1995

Performance evaluation of an auction based manufacturing system using generalized stochastic Petri nets.

Shubhabrata. Biswas
University of Windsor

PERFORMANCE EVALUATION OF AN AUCTION BASED MANUFACTURING SYSTEM USING GENERALIZED STOCHASTIC PETRI NETS

By

Shubhabrata Biswas

A Thesis
Submitted to the Faculty of Graduate Studies and Research
Through the Department of Industrial and Manufacturing Systems
Engineering in Partial Fulfilment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

1995

ISBN 0-612-01431-2

Canada

Name SILVIA ABRATA BISCAS

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

INDUSTRIAL
SUBJECT TERM

0546
SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984
Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

ABSTRACT OF THE THESIS

Auction Based Manufacturing is a form of Heterarchical control. Such a system relies upon the intelligence of individual machining centres to allocate parts to prospective machining centres which can do the next stage of machining. This approach eliminates the requirement of a costly pre process scheduling run, which can prove to be inappropriate in the event of unforseen machine breakdowns or changes in machining capacities. Such a manufacturing system is capable of automatically adjusting itself to changes at the shop floor. The schedule is generated on a Real Time basis based on the process plan for each Part Type and the available machines at that instant of time.

While Auction Based Manufacturing has significant strengths, it also has some drawbacks. One of the major drawbacks is the lack of predictability of the exact behaviour of the shop floor. It is perceived that a method by which an operator can predict the most likely behaviour of the system is highly desirable in order to fine tune the control algorithms or rules to achieve optimality and make the right resource available at the right time for the right amount of duration .

This thesis deals with the modelling and the performance evaluation of an Auction Based Manufacturing system that has multiple machining stages, multiple part types and multiple parts. Each part type has a process plan with multiple stages, further each stage can have multiple alternatives available. Four different models have been analyzed to understand the impact of control heuristics, tie-breaking rules, number of parts and machining rates on the overall system behaviour. The models have been studied using both steady state and time based transient analysis techniques. A technique by which the most expected behaviour of such a system can be predicted has been suggested. The tool used for the analysis is Generalized Stochastic Petri Nets. The Stochastic Petri Nets Package (SPNP) ver 3.1 was used as a solver for the Petri Net models. The models have been analyzed using performance evaluation parameters like machine utilization value, throughput level and expected value of queue length.

To My Parents

Rathindranath and Bithi Biswas

and My Brother

Ritwik Biswas

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Dr. S.P. Dutta for his valuable advice, guidance and support throughout the duration of my research. I would like to thank my committee members Dr. M. Ahmadi of Electrical Engineering and Dr. M.H. Wang for their constructive criticism and guidance during my research. I would like to mention a special word of thanks to Dr. Myron Hlynka of the Mathematics Department for his timely advice and helpful suggestions on the theoretical aspects of my research work. My sincere thanks goes to Ms. Jacquie Mummery for her friendship, encouragement and help throughout my masters program. Her cheerful personality and witty humour went a long way in making the masters program all the more enjoyable. I am indebted to Mr. Tom Williams for his advice and his help in running the softwares and the computers that I used during my research.

I am grateful to Dr. Christoph Lindemann of the Technical University of Berlin for generously providing us with the software DSPNexpress1.3 which I used during certain stages of the modelling process.

I would finally like to thank all my colleagues and friends at the school for their wonderful company and for the fun filled times we shared together. I would especially like to thank Michael Johnson, Abi Philipose and Albert Hui for their wonderful company and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AGV	: Automatically Guided Vehicle
AMS	: Automated Manufacturing System
ASRS	: Automated Storage and Retrieval System
CNC	: Computer Numerically Controlled
CTMC	: Continous Time Markov Chain
DEDS	: Discrete Event Dynamic System
DSPN	: Deterministic and Stochastic Petri Net
FMS	: Flexible Manufacturing System
GSPN	: Generalized Stochastic Petri Net
HSPEED	: High Speed
MC	: Machine
NC	: Numerically Controlled
SPN	: Stochastic Petri Net
SPNP	: Stochastic Petri Net Package
SQL	: Shortest Queue Length
TFIRST	: Take First
TLAST	: Take Last
TPN	: Timed Petri Net
TPPN	: Timed Place Petri Net
TTPN	: Timed Transition Petri Net

Chapter 1

INTRODUCTION

1.1 Background of Problem

An automated manufacturing system is a complicated Discrete Event Dynamic System (DEDS) that consists of several major entities. They are the ASRS system, the machining centres, the AGV's, the communications network, the database and the palletized parts. It also consists of a control structure which determines how the resources in the system are allocated with respect to time in order to achieve the ultimate goal of getting finished parts in the shortest possible time and with the highest resource utilization.

There has been a school of thought for the past decade that feels that the authority for resource allocation should be delegated to the components of the AMS themselves and not restricted to a supervisory controller. This concept professes the idea of having intelligent entities in the AMS, in the form of pallets, AGV's and machines which can communicate with each other and take decisions based on heuristics and guided by the immediate present status of the system on a real-time basis. One such approach is

Auction-Based manufacturing. This approach has been found to have numerous advantages.

It tries to do away with complex and time consuming scheduling calculations that are required by the traditional approach for a set up having multiple part types and alternative process plans. Such calculations while being quite complicated in themselves, also often need to be repeated all over again even with the slightest change in the shop floor status eg. machine breakdown or production at half capacity. Real-time intelligent scheduling as exemplified by Auction-Based Manufacturing tries to approach this problem from a fresh perspective and offer solutions.

A shop floor based on auction based manufacturing is capable of automatically adjusting itself to the changing scenario on the shop floor. Whenever a machine breaks down or becomes unavailable, the system automatically stops allocating parts to such a machine. Also, there is no necessity to build up an elaborate schedule describing machine-part allocation in advance. In Auction Based Manufacturing, the schedule is generated on a Real Time basis based on the process plan for each Part Type and also on the available machines at that instant of time. No prior assumptions are made as to the availability of machines; hence the system is insensitive to unexpected changes on the shopfloor. Such a system only requires the operator to specify the process plan along

with alternatives for each part type, the number of units of each and the time each part type is required to spend on each process stage. The system can be then be loaded up and left to run autonomously.

While Auction Based Manufacturing has significant strengths in terms of being Real Time, Robust and Autonomous, it also has some drawbacks. One of the major drawbacks is the lack of predictability of the exact behaviour of the shop floor owing to total autonomy of the system. It is perceived that a method by which an operator can predict the most likely behaviour of the system is highly desirable in order to fine tune the control algorithms or rules to suit optimality or specific shop floor requirements. There has been some work done in modelling of AMS's based on auction-based manufacturing in the past ten years but not many numerical results are available. This thesis attempts to analyze AMS's based on the heterarchical control in detail and presents results of their performance evaluation. It also suggests a technique by which the most expected behaviour of the system can be predicted.
From the literature survey, the details of which have been provided in Chapters 2 and 3, the author is not aware of any work that addresses this issue directly.

The tool used in this study for modelling and simulating the AMS is Petri Nets. They are particularly useful for the modelling and analysis of large Discrete Event Dynamic

Systems (DEDS) which display characteristics like concurrency, asynchronization, conflicts, mutual exclusion etc. They also offer a simplistic and visually perceptible method for defining an AMS. The form of Petri Nets used in this research is the Generalized Stochastic Petri Nets. This type of Petri Nets has the added advantage of being able to handle exponentially distributed time delays in a DEDS.

1.2 Statement of objectives

1) To model an enhanced version of an AMS consisting of multiple machining cells, multiple part types, multiple parts within each part type and separate process-plans for each part type with in-built alternatives.

2) To study the time based transient behaviour and the steady state behaviour of the system based on variation in control heuristics and system parameters like machining rates and number of parts.

3) To try to develop a methodology to predict the most likely behaviour of an Auction Based Manufacturing System based on its real time analysis and long term steady state analysis results.

1.3 Organization of Thesis

This thesis has 7 Chapters. Chapter 2 discusses the pros and cons of both hierarchical control and heterarchical control to show the motivations for investigating Auction-Based manufacturing systems. A literature survey accompanies the Chapter contents.

Chapter 3 discusses the application and theory of Petri Nets with respect to the modelling and analysis of Automated Manufacturing Systems. The focus is on Generalized Stochastic Petri Nets (GSPN). Literature survey carried out in this regard is also given in this chapter.

Chapter 4 discusses the mathematical basis for performing the steady state and the time based transient analysis of Petri Nets. These concepts have been further explained with the help of simple examples.

With this background of Heterarchical control and Petri net theory, Chapter 5 discusses the Petri Net models that have been developed and there method of analysis. It discusses the basic and the common features of the models and the individual variations between them. It also describes the performance evaluation parameters that are used to evaluate the system.

Chapter 6 discusses the results obtained and presents observations and analysis of the results.

Chapter 7 is a compilation of all the lessons learnt through this research work. It records the computational experience and other technical details about which

information was gathered during the course of this research. It has the concluding remarks and discusses the scope for future work in this directions. New avenues that could be explored have been suggested. It also suggests ways by which this work could be integrated with other available work to develop more comprehensive and integrated decision support systems.

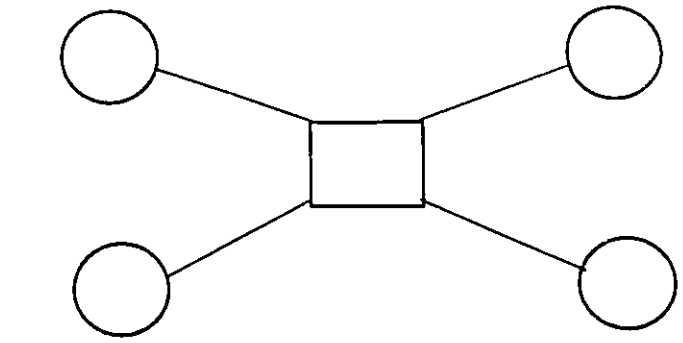
Chapter 2

HIERARCHICAL CONTROL vs. HETERARCHICAL CONTROL

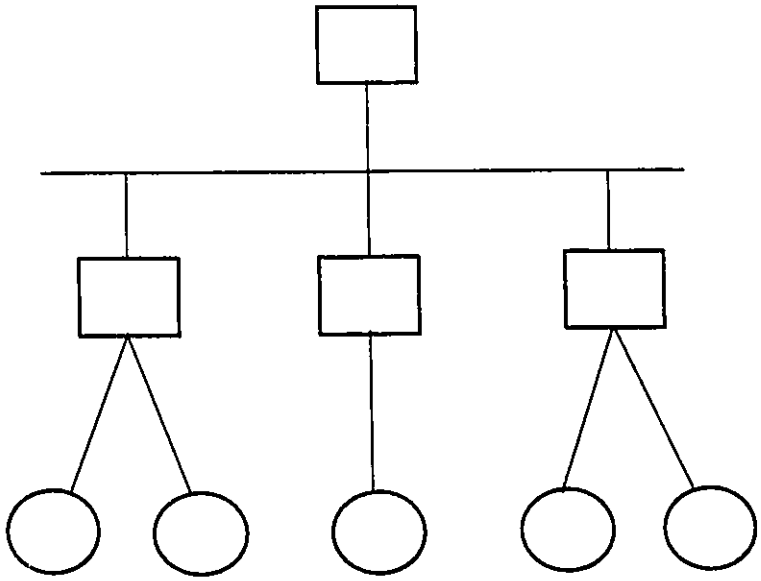
2.1 Introduction

The control system of an automated manufacturing system (AMS) coordinates and directs the parts handling and processing activities that transform raw materials into finished products. As a result, the control system embodies many decision making responsibilities including part scheduling, part routing and resource allocations within the manufacturing facility. A control architecture creates a control system from components. It allocates these decision making responsibilities to specific control components and determines the interrelationships between the control components. For example, a control architecture may specify one control component with responsibility for part scheduling, and another for the movement of parts. Also, the interaction between the two components can be stipulated.

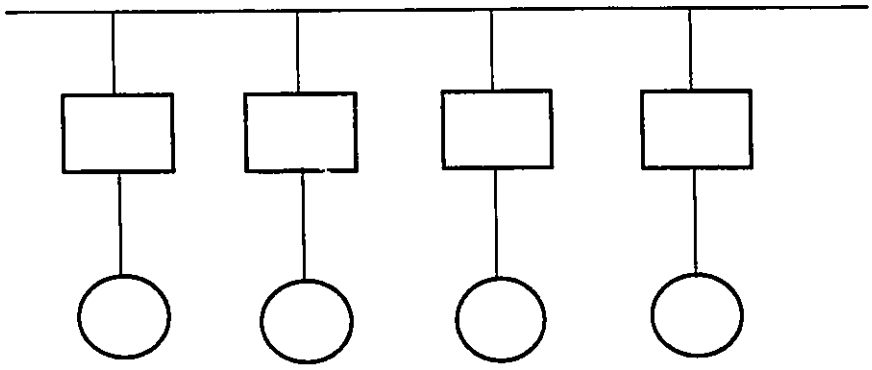
The earliest control architecture employed a centralized approach. With the development of computer technology and advanced manufacturing facilities, control architectures for manufacturing systems have evolved from their traditional centralized



a) Centralized form



b) Hierarchical form



c) Heterarchical form

Fig 2.1 Basic forms of control architecture

forms, to the hierarchical form, and recently to the heterarchical form, to meet the demands of AMSs: reliability/fault-tolerance, modifiability/extensibility, and reconfigurability/adaptability, (Dilts et al.,1991). Fig. 2.1 shows the structure of the three basic forms of control architectures. The hierarchical control strategy is the most commonly applied method nowadays. However, due to its increasing complexity with system size, hierarchical control method has been challenged recently by the heterarchical control approach. In this chapter, a comparison between hierarchical control and heterarchical control is presented to show the motivations of this research.

One of the major drawbacks of hierarchical or centralized control is that it limits the systems' short and long term flexibility. Unforeseen disturbances cannot be handled easily and every time a new set of products is to be produced, a new resource coordination system has to be developed. Changes in the configuration of the production system, e.g. introduction of a new machine, or a machine breakdown, require global changes in the coordination software. The introduction of resource coordination mechanisms, where instead of rigid hierarchical relations, heterarchical relations are established dynamically, seems to offer a new boost to the time based competitiveness of the modern manufacturing enterprise, (Upton,1992;Yang et al.,1993). One proposed method is to have intelligent machines in the system. The intelligent machines after producing process plans for the product and after examining

their state (including buffer state, tool-magazine state etc.) provide bids for the processing of the jobs. The job is given to the best bid.

2.2 Hierarchical control strategy

The hierarchical control architecture is characterized by a "philosophy of levels" of control and contains a number of control modules arranged in a pyramidal structure, (Duffie et al.,1988). Rigid master/slave relations exist between decision making levels with the control decisions operated top-down and status reported bottom-up.

A Flexible Manufacturing System (FMS) is typically a system of hierarchical control. The FMS has become the focus of attention in industry and in academic research for more than a decade. Numerous FMS have been implemented and they have brought tremendous benefits in reduced inventory, improved quality, and shortened lead time. However, there are some problems with the traditional FMS, which prevents it from being widely implemented. The major reasons are that it lacks long-term flexibility and is limited in size because of the complexity of the control software.

1) FMS Software Complexity

An FMS is an integrated, computer-controlled complex of automated material handling devices and numerically controlled (NC) machine tools that can simultaneously process medium-sized volumes of a variety of part types. The goal of FMS is to attain the

efficiency of flow shop and the flexibility of job shop.

In general, flexibility is the speed at which the system can react to and accomodate changes. In FMS, flexibility can take a number of forms, including, (Pimental,1990):

a. Volume flexibility: The ability to handle changes in the production volume of a part or finished good.

b. Routing flexibility: The ability to route parts through the system in a dynamic fashion taking into account machine breakdowns, required tooling and other necessities.

c. Product flexibility: The ability to handle requests for a wide variety of products, including the ability to reconfigure the system to handle the production of different products. The key idea in FMS is that the coordination of the flow of work is carried out by a central control computer, and usually a hierarchical approach is applied to accommodate real-time control. Therefore, to implement the flexibility mentioned above, the FMS control software would be very complex and expensive, although global optimization can be accommodated.

2) Modifiability and reconfigurability: The hierarchical control structure tends to be rigid and fixed in the early design stages for a range of known products under given configurations. If, over the long term, new families of products are introduced into the system and new machines are added (or an old one updated), the manufacturing system must be shut down in order to implement the software changes, and the modification

would be very costly to make because of the complexity of the central control software. For the same reason, it is also difficult to accommodate the addition or removal of various manufacturing system components while the system is operating.

3) Fault-tolerance: Because of the master/slave relationships between the control levels, a failure of a master controller at a higher level will cause the failure of all slave controllers at the lower levels. The higher the failed master controller, the greater the number of paralysed slave controllers.

4) Limited system size: If a large shop consists of 100 or 200 CNC machines, it would not be economically controllable with traditional centralized FMS because of the complexity of the control software. In other words, FMS are limited in size.

2.3 Heterarchical Control strategy

2.3.1 Principles of Heterarchical Control

To overcome the shortcomings associated with the traditional centralized and hierarchical control structures, a new strategy, called "heterarchical control" has been proposed by several researchers. Although there are some differences among the proposed methods, the basic ideas are the same. That is, the central control computer

is eliminated and all the system entities are fully autonomous, making decisions locally through negotiation (auction) process.

In general, the negotiation processes are divided into four phases:

1) part posting bid request,

2) workstation evaluation and bid generation,

3) part acceptance and commitment, and

4) workstation confirmation

Fig. 2.2 shows one part, one automated guided vehicle (AGV), and three machining centres (MCs). Part A broadcasts a request for an operation to all the machining centres in the system. One possible case will be that two of the machining centres are free and capable of doing the job, therefore, both respond. Part A selects the best bid with the lowest throughput time,say,that of MC#3, and makes the reservation. MC#3 confirms the reservation by sending an acknowledgment. Then part A arranges a vehicle for its transportation. Another possibility is that none of the machines is free. In this case, part A has to wait a certain period of time (back-off time) and calls again. In real-world situation, there could be multiple parts in a system undergoing the negotiating procedure simultaneously.

The advantages of heterarchical control strategies are as follows:

1) **Reduced software complexity**: From the example in Fig. 2.2, it can be seen that a part has been processed without a central control computer, but with simple, modular, physically decentralized hardware and software.

2) **Reconfigurability/adaptability**: When a new machine is added into the system, t here is no need to take down the system. The new machine may simply be told the "rules of the game" and becomes a part of the system with no lost production. Removing a machine from the system is also straightforward. It simply stops bidding, and jobs stop coming to it.

3) **Fault-tolerance**: If a machine is down, it stops responding to requests. So failures are limited to the locale where they occur so that the system-wide consequences are avoided. In other words, the fault tolerance is implicit in the design.

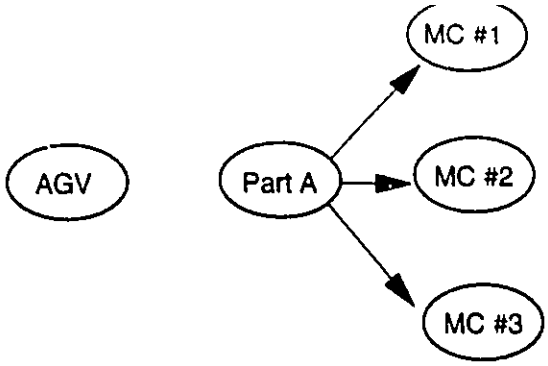
4) **Larger system size**: As many machines as needed can be added as long as the network is not saturated.

Meanwhile there are disadvantages of heterarchical control which are mainly two fold:

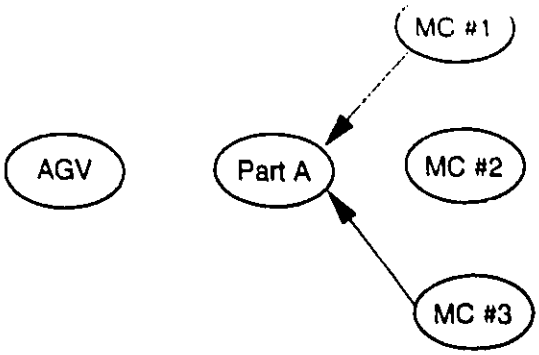
1) **Local optimization**: From the myopic bid structure, it can be seen that local decisions made by entities are, of course, not globally optimal. For example, a part may opt for a machine that is the most appropriate in the short term, but may find itself away from its optimal total processing path from a long term point of view.

2) **Enabling technologies**: The technologies associated with this type of production systems, such as, the chip on a part, inexpensive radios, cheap computers, communication networks with high capacities, and automatic process planning, need to be further developed.

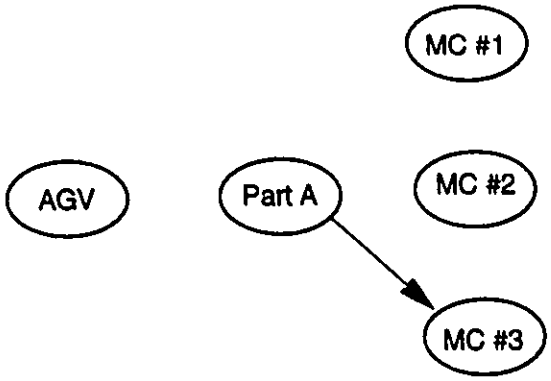
Although the heterarchical control architecture is somewhat restricted by the limitation of existing hardware and software technology, there appears to be a growing acceptance of the concept by academics and industry leaders. Table 2.1 is a summary of comparison between hierarchical and heterarchical control.



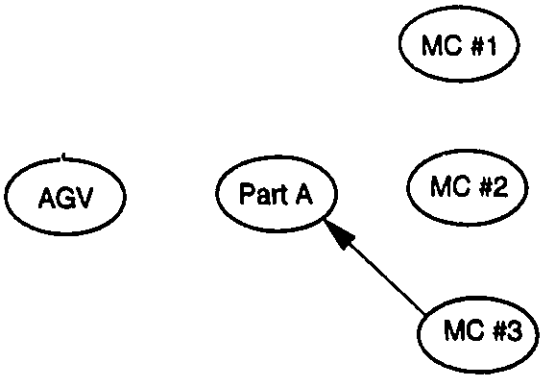
a) Part A broadcasts a request



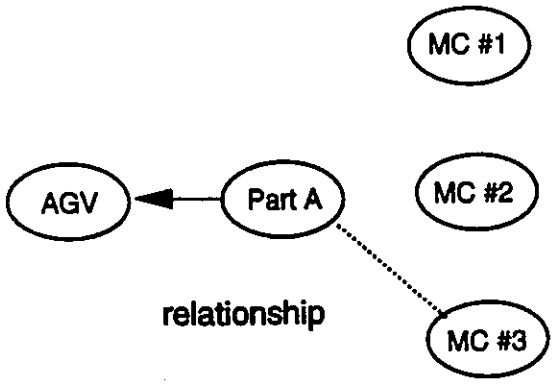
b) Two machines are free to respond



c) Part A sends reservation to MC #3



d) MC #3 acknowledges Part A's reservation



e) Part A arranges transportation

Fig 2.2 Negotiation process of heterarchical control

Table 2.1 Comparison between hierarchical and heterarchical control

Characteristics	Hierarchical control	Heterarchical control
Control architectures	rigid master/slave relations between decision making levels	-no master/slave relationships; -full local autonomy
Software complexity	very complicated	reduced
Modifiability/extensibility	difficult	easy
Reconfigurability/ adaptability	difficult	easy
Reliability/fault-tolerance	The higher the failure nodes,the greater the number of lower paralysed controllers.	-local failure has no system wide influence. -implicit in the design
Optimality	global	local
Communication capacity required	lower	higher
Enabling technology	implemented already	under development

2.3.2 Literature survey

The need for an entirely new approach to the design of systems, which can be changed, adapted, expanded or updated as the situation requires, was recognized as early as in 1978 by Hatvany. Cooperative hierarchies were suggested by Hatvany later in 1985. (Smith,1980) developed a contract negotiation scheme for cooperative problem solving, providing a starting point for the procedures in the "heterarchical" control architecture.

Since then, work has been carried out to provide feasible manufacturing operations based on the negotiation models in a variety of circumstances.

(Lewis et al.,1982) developed a data flow of computerized manufacturing systems. Their pioneering work attempted to overcome centralized scheduling complexity in flexible manufacturing.

(Shaw and Winston,1985) described a negotiation protocol used to ensure orderly information transformation and events sequencing between asynchronous, cooperating manufacturing cells. Petri nets were used as a modelling language for executing the task sharing process.

(Fukuda et al.,1986) briefly described a hierarchical yet distributed control approach to overcome dynamic variations in manufacturing. They introduced a common message board to permit information exchange between control modules.

(Maley,1988) proposed a system called CADENCE (Computer Automated Distributed Environment for Network Coordination and Execution), in which both the physical flow dependencies and the information flow were provided by utilizing a negotiation algorithm. CADENCE went beyond previously proposed task bidding structures by utilizing distributed decision making in managing the flow of individual intelligent parts.

(Parunak et al.,1985) utilized the actor model to meet the complexity issues in manufacturing. Their distributed system worked in a hierarchical framework but they

reported that initial studies in lateral negotiation showed promise. A system called YAMS, has been designed to implement Smith and Davis' contract net in a manufacturing environment, (Parunak,1987).

A heterarchically controlled machining cell has been constructed at the University of Wisconsin-Madison, (Duffie and Piper,1986). Initial results indicated that the heterarchical approach had attractive attributes for control of flexible manufacturing systems and cells in lower development costs and improved modifiability.

Three flexible machining cell controllers have been implemented later by (Duffie and Piper,1987) in an effort to analyze the relative advantages and disadvantages of hierarchical and heterarchical cell control architectures. Results showed that the heterarchical approach possessed a number of advantages including increased fault-tolerance, inherent adaptability and reconfigurability, decreased complexity, and reduced software development cost.

An experimental heterarchically controlled manufacturing system has been developed which consists of a robotic machining cell and a robotic assembly cell, (Duffie et al. ,1988). A heterarchical control architecture and a set of underlying design principles for developing and implementing such a system was emphasized.

Although various schemes have been proposed and experiments were carried out, not many numerical results were reported until the work published by (Upton et al.,1991, 1992). A heterarchical control structure was investigated for a large manufacturing

system which was prone to failure and change. The difficulties in building queuing models for auction-based manufacturing were discussed. To get some insights into the behaviour of these systems, some numerical results, such as, machine utilization, number of rejection and auctions won vs. arrival intensity were obtained by simulation models.

Chapter 3

PETRI NETS

3.1 Introduction

The traditional methods for analysis of Automated Manufacturing Systems fall into two categories: analytical methods (including queuing networks and mathematical programming models) and simulation models. Recently Petri nets, especially Generalized Stochastic Petri Nets, have appeared as promising tools for performance analysis. Of course each approach has its own advantages for system modelling. In this section, a comparison between analytical, simulation and the Petri Net models is made to show why Petri Nets were chosen as the modelling tool for this research.

Analytical methods use standard solution techniques and provide optimal solutions. However, restrictive assumptions are usually made in order to achieve analytical tractability. For example, by assuming that the customer interarrival time and server service time are exponentially distributed and independent of system state, the Product Form Queuing networks are computationally very easy to solve. Nevertheless the modelling power of these Queuing Networks is low:Priority , blocking and finite buffer

size are difficult to model; complex layouts, synchronization and control policies for despatching customers typically cannot be handled. In addition it is very difficult to get an intuitive understanding from the formulae which sometimes runs into several pages. As a result, the method is usually used at the preliminary design stage to distinguish between alternative designs on an aggregate basis.

Discrete event simulation enables detailed description and analysis of system behaviour and is useful in all stages of design and operational analysis. However, the model building process can be expensive and time consuming. Moreover, in simulation, complex interactions cannot be understood visually although it can model a complex system at any detailed level.

Petri nets are modelling tools that lie between analytical and simulation methods, providing analytical results with much of the modelling flexibility of simulation. Petri nets are representatively powerful because they allow model description in a conceptually simple and graphical manner; They can exactly model non-product form features, such as priority, synchronization, blocking, splitting of customers etc. The modelling power is also enhanced by their capabilities in analyzing performance quantitatively as well as in verifying models qualitatively. They are computationally efficient because the mathematical foundation of Stochastic Petri nets is Continuous

Time Markov Chains (CTMC), but the construction of the entire state space is avoided since this process has been automated by some software packages.

The major assumptions of the Stochastic Petri Net approach is exponential distribution of interarrival and service time, which are widely used in analytical and simulation modelling. The limitations lie in the net structure and the net reachability tree could be very complex in a large system. Coloured Petri Nets can lead to compact representation of nets, but the solution complexity is the same as the unfolded model.

In conclusion, Petri Net modelling is most suited when the researchers try to understand a system with synchronization and cooperation among concurrent processes. Table 3.1 shows a comparison between queuing, simulation and Petri net models.

The manufacturing system under investigation has the following characteristics:

- Concurrency or parallelism: Many operations take place simultaneously.
- Conflict: More than two processes may require a common resource, such as buffer or a machine, at the same time.
- Synchronization: The starting of machine operations must be controlled to ensure correct operation of the entire system.

- Splitting customers: Dynamic decision making is evolved for dispatching of parts.
- Blocking: Because of finite buffer sizes and limited capacity of the machining centres, a part may be blocked when its first operation is finished.
- Priority: Different events may have different priorities.

For the above mentioned reasons it is clear that queuing models cannot be applied. Simulation can be used, but we prefer to use Petri Nets because we understand the task allocation policy conceptually and theoretically.

Table 3.1 Comparison between Queuing, Simulation and Petri Net Models

		Queuing models	Petri net models	Simulation models
Assumptions		exponential arrival rate and service time which are independent of system state	exponential timed transitions	any distribution
Modelling powers: synchronization, concurrency, control policies, priority, blocking, finite buffer size		low or no	higher	highest
Intuitive understanding		difficult	easy	difficult
Solutions model	model verification	no	yes	no
	performance evaluation	easy	easy with problems of reasonable sizes; impossible when a state space explodes, but a verified PN model serves as a model for simulation.	expensive (time-consuming), difficulties in output interpretation, bugs undetectable
Application scenarios		at the initial design stage when resource allocation mechanisms are not considered	at all design stage; most efficient for issues about synchronization, complex resource allocation etc.	at all design stages

3.2 Applications of Petri Nets to Manufacturing and Automation

3.2.1 Modelling and Validation

In order to explore the applicability of Petri nets for flexible production systems, (Valette et al..1982) has specified and validated interconnected controllers for a transportation system in a car production system using Petri nets. Their experience show that Petri nets are applicable to this system and indicate that such a Petri net approach can be based on decomposition and structuring. Furthermore, (Valette,1987) argues that Petri nets are more convenient than other models for concurrency. Examining the same production system used by (Valette et al. 1982), (Alla et al.,1985) employed coloured Petri nets to model the system. Their coloured Petri net model has been proven to be deadlock-free and bounded. In the same paper, the possible benefit of using coloured Petri nets over ordinary Petri nets has been revealed; i.e. their conciseness makes it possible to describe a complex FMS with many similarities.

To fully explore the advantages and disadvantages of ordinary Petri nets and coloured Petri nets, (Martinez et al.,1986) modeled a transfer line consisting of two machines and three buffers, as well as a transportation system consisting of an automatic guided vehicle (AGV), three loading and one unloading stations, using both nets. They briefly discuss the characteristics of a "good model", including boundedness, liveness, and

reversibility. Other specific properties include mutually exclusive places and finiteness of synchronic distance.

To understand behaviour such as deadlocks and buffer overflows of an FMS, (Narahari and Viswanadham,1985) modeled two manufacturing systems; a transfer line with three machines and two buffers, and an FMS with three machines and two part types. Boundedness and liveness were analyzed for both systems using invariant methods, and the significance of boundedness, liveness, and reversibility in manufacturing systems was explored. Furthermore, they demonstrate the usefulness of coloured Petri nets in automated manufacturing systems, (Viswanadham et al.,1987).

To facilitate Petri net modelling and to allow more flexibility, some extended Petri net models have been proposed. For example, structured adaptive and structured coloured adaptive Petri nets include inhibitors, self-modifying arcs, etc., and their usefulness is illustrated through design of an FMS control system, (Gentina et al.,1987).

A marked graph is decision-free since each place has only one output transition. For ordinary Petri nets, a place can have many output transitions; as a result, different choices in firing enabled transitions may exist. To resolve such conflict situations, (Krogh and Sreenivas,1987) added NOT places in a net model to eliminate ambiguity in firing enabled transitions. The net becomes essentially decision free; thus, there is no ambiguity in realtime resource allocation for manufacturing processes.

To explore system behaviour and to verify logical correctness of specifications, (Wang

et al.,1989) modeled Connection Management Services of the Manufacturing Message Specification (MMS) using coloured Petri nets. The resulting net model offers a hierarchical mathematical description for the entire MMS protocol and can be used for performance analysis for the protocol.

(Wang,1993) used Object Oriented Petri Nets to model the shopfloor in a modular fashion. The control approach used for this work is based on the hierarchical approach. The shop floor entities are divided into physical objects, information objects and control/decision objects. Information is inherited and shared amongst these objects in a way similar to the objects in object oriented programming approach.

Petri net models often tend to have numerous subnets interacting together and can be quite cumbersome to handle. To introduce modular structuring and improved modifiability of the model, (Inamasu and Porto,1993) have discussed an object oriented approach for linking Petri net models together.

(D'Souza and Khator,1994) have compiled a wide range of information on the analysis and performance evaluation of AMS's using Petri nets. They have discussed a wide range of Petri Nets including Timed Petri Nets (TPN), Stochastic Petri Nets (SPN), Generalized Stochastic Petri Nets (GSPN) and Deterministic and Stochastic Petri Nets (DSPN) in the context of deadlock analysis and control of AMS.

Based on Petri net theory, software packages have been developed for automatic validation of operations in FMS, (Colom et al.,1986,1987); (Martinez,1987); (Crutte

and Gentina,1992) have applied Petri nets for design and validation of operation sequences in FMS and (Ezpeleta and Martinez,1992) performed formal specification and validation for production plants using high-level Petri nets.

Barson,1993) introduced a new Petri Net based tool called UNISON which uses coloured place-transition nets. It is quite a comprehensive tool and can be used for system design, modelling, validation and simulation. One of the strengths of this tool is that along with regular modelling and validation it can also be used during run time to run external programs and functions that call hardware or and external program, eg. a communication system.

(Van der Aalst,1994) has sited a wide range of industrial applications where High Level Petri Nets are being applied. He has sited examples of actual Petri Net applications in Software prototyping, Logistics and Resource allocation in Manufacturing, Administration of Organizations and Traffic Control.

3.2.2 Performance Analysis

As previously mentioned, systems were modeled using timed Petri nets in order to conduct temporal performance analysis, i.e., to determine production traits of systems, resource utilization, and the like. It is also possible to detect a bottleneck in an FMS or to determine optimal buffer size, optimal pallet distribution, etc. as indicated by (Dubois and Stecke,1983). In fact, Dubois and Stecke were the first to apply timed Petri

nets to describe, model and analyze production processes. In their research, deterministic time variables were assumed, and a Petri net-based simulation method was utilized to find minimum cycle time and to identify the bottleneck machine for an FMS with three machines and three part-types in a fixed route.

In the 1985 International Workshop on Timed Petri Nets, many research projects in the field of timed Petri nets were presented, and various applications were reported for analyzing the performance of computer systems, (Holliday et al.,1985), communication protocols, (Gressier et al.,1985), and manufacturing systems, (Bruno et al.,1985). More importantly, some significant software packages were introduced, for example DEEP, (Dugan et al.,1985), and ESP, (Cumani et al.,1985). (Chiola ,1985) presented a user-friendly software package for analysis of generalized Petri Nets, whose improved version is called GreatSPN. GreatSPN is powerful in the sense that it can accept various time variables, inhibitors, and random switches and has simulation capacity.

To evaluate the performance of job-shop systems under deterministic and repetitive functioning of a production process, (Hillion and Proth,1989) applied a special class of timed Petri nets called timed event-graphs for an FMS with three machines and three job-types. The number of jobs in-process was nearly minimized using integer programming, while the system still worked at its maximum productive capacity.

(Viswanadham and Narahari,1988) have provided an excellent introduction to the use

of generalized Stochastic Petri nets (GSPN) in analyzing the system performance of automated manufacturing systems. They use a software package they developed to evaluate two representative systems: a manufacturing cell with multiple material handling robots, and an FMS with three machines and two part-types.

All these existing software tools can be used to analyze the performance of automated manufacturing systems. For example, (Al-Jaar and Desrochers,1990) investigated the performance of transfer lines and production networks using SPNP developed by (Ciardo,1989). (Zhou and Leu,1991) utilized SPNP to evaluate the performance of a two-robotic manipulator station for printed circuit boards.

Aiming at demonstrating applicability of and accuracy achieved by stochastic Petri nets for FMS, (Watson and Desrochers,1991) performed three representative case-studies from different sources using SPNP and GreatSPN. Their work shows that the performance analysis results obtained using Petri nets agree with those obtained by simulation, (Dunkler et al.,1988), queuing theory, (Yao et al.,1985), and probability theory, (Seidmann et al.,1989).

SPNP continues to be used for analyzing the performance of manufacturing systems which have fixed routing and produce limited number of parts. (Zhou et al.,1990) first developed a stochastic Petri net modelling approach such that the good behavioral characteristics of the model could be preserved. Then they analyzed deadlock-free and deadlock-prone manufacturing systems. Their results show that supervisory controllers

with freedom from deadlock are better than deadlock-prone controllers in designing real-time resource-sharing distributed systems.

Recent developments in performance analysis of automated manufacturing systems has led researchers to investigate FMS with possible deadlocks, (Viswanadham,1990). One reason for this investigation is that it can be very difficult to design a deadlock-free controller for complex systems with numerous job types.

Moment generating functions and Petri nets have been combined to obtain symbolic performance solutions for stochastic Petri nets, generalized stochastic Petri nets, and extended Stochastic Petri nets, (Guo et al.,1991); (Zhou et al.,1992). Reduction methods have been used to derive the closed form of the performance indices, (DiCesare et al. ,1992).

In order to explore the steady state system behaviour under different combinations of machine selection, job selection and vehicle dispatching rules (Raju and Chetty,1993) used priority nets to model an FMS consisting of AGV's, multiple part types and multiple machining centres.

Results from the steady state analysis of Petri net models of an AMS has been used for capacity level planning of the shop floor entities like AGV's, (Chin,1993). The model was developed using GreatSPN 1.5.

(Ramaswamy and Valanavis,1993) have used Extended Petri Nets (EPN) to model, analyse and simulate soft failures in a real time material handling system.

The performance evaluation of an Auction Based Manufacturing system has been performed using steady state analysis results for a multiple machine, multiple part type AMS (Zhou et al.,1994).

Although the underlying models of Stochastic Petri nets are still Markov processes, the Petri net approaches preclude the direct construction of the state space required by Markovian analysis. This is often too large for humans to manipulate. In order to deal with a stochastic net of a large state space, reduction and approximation methods have been investigated, (Jungnitz et al.,1991,1992); (Ma et al.,1992). Some reduction techniques for ordinary Petri nets have been extended to the generalized stochastic Petri nets to make it possible to analyze the performance of large-scale discrete event systems. Queuing network methods and GSPN have been combined to model and analyze complex systems, (Balbo et al.,1987,1988). A decomposition method has been developed to evaluate SPN models and applied to a manufacturing system, (Ciardo et al.,1991).

3.3 Petri Nets vs. other Models in Modelling and Control

An event-driven system can be abstracted as a state machine in which the states change when events occur. The finite state machine or automaton results when the total number of states in a system is finite. Compared with other models discussed, they have the following advantages, (DiCesare et al.,1991); (Ma et al.,1992); (Martinez et al.,1986); (Zhou et al.,1989):

1. Ease of modelling DES characteristics: concurrency , asynchronous and synchronous features, conflicts, mutual exclusion, precedence relations, non-determinism, and system deadlocks.

Excellent visualization of system dependencies

Focus on local information

Top-down (step-wise refinement) design

Bottom-up (modular composition) design.

2. Ability to generate supervisory control code directly from the graphical Petri net representation.
3. Ability to check the system for undesirable properties such as deadlock and instability and to validate code by mathematically based computer analysis-no time consuming simulations for many cases.
4. Performance analysis without simulation is possible for many systems. Production rates, resource utilization, reliability, and performability can be evaluated.
5. Discrete event simulation that can be driven from the model.
6. Availability of status information that allows for real-time monitoring.
7. Usefulness for scheduling because the Petri net model contains the system precedence relations as well as constraints on discrete event performance.

As a single representation tool as shown in Fig. 3.1, Petri nets can aid in modelling, analysis, verification, simulation, scheduling, and performance evaluation at design stage. Once the system shows desirable behaviour, the net can be converted for control and monitoring operations at run time. Therefore, Petri nets can be regarded as a powerful mathematical and graphical tool for design of various discrete event systems.

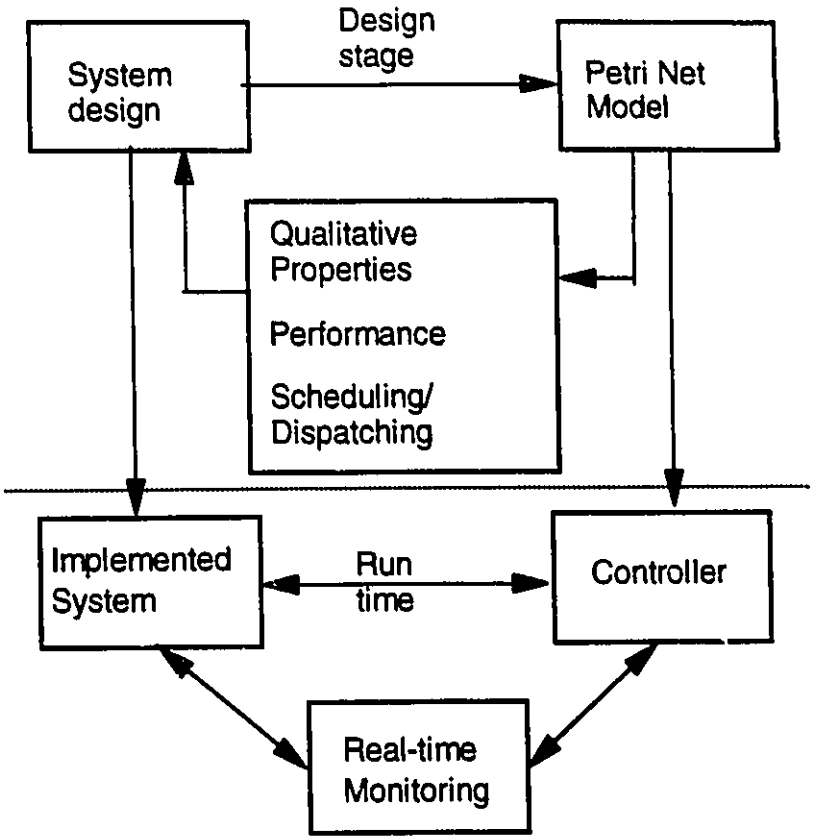


Fig 3.1 Petri Nets as a single representation for modeling, analysis and control

3.4 Definition of Petri Nets

There are variations among the available definitions of Petri nets. However, the differences are notional. The format given by (Ajmone et al.,1984) is followed in this research.

3.4.1 Standard Petri Nets

Definition 1: A *marked Petri net* is a five tuple

$PN = (P, T, A_i, A_o, M_o)$

where

$P=\{p_1, p_2, ..., p_n\}$ is a set of *places*,

$T=\{t_1, t_2, ..., t_m\}$ is a set of *transitions*,

$A_i \subset (P \times T)$ is the set of *input arcs* that defines directed arcs from places to transitions,

$A_o \subset (T \times P)$ is the set of *output arcs* that defines directed arcs from transitions to places,

$A_o = (A_i \cup A_o)$ is the set of transition arcs, and

$M_o = \{m_{o1}, m_{o2}, ... m_{on}\}$ is the *initial marking*

In the graphical representation of a Petri Net (PN), places are drawn by circles and transitions by bars. The input and output places are represented by directed arcs from places to transitions and vice versa, respectively, if arcs exist between them. Tokens are denoted by black dots residing in places.

In a manufacturing system, a place is usually a shared resource or a condition for an event to occur. A transition is generally used to represent the initiation or termination of an event. Input functions and output functions establish unidirectional relationships between places and transitions, and transitions and places, respectively. Token in places indicate that resources are available or conditions are true.

Definition 2: A marked Petri net executes according to the following rules:

1) A transition is enabled when all its input places contain at least one token.
2) An enabled transition can fire thus removing one token from each input place and placing one token in each output place.
3) Each firing of a transition modifies the distribution of tokens in the places of say marking M, and thus produces a new marking called M′. M′ is called immediately reachable from M.

Definition 3: The reachability set of a Petri Net $R(M_o)$ is defined as the set of all markings that are reachable from the initial marking M_o.

As Standard Petri Nets do not include the concept of time, they are used only for qualitative and a logical analysis of systems. The emergence of Timed Petri nets (TPN) made it possible to perform quantitative analysis of systems. By associating a constant time delay to either places or transitions, Timed Place Petri Nets (TPPN) and Timed

Transition Petri Nets (TTPN) were created. Stochastic Petri nets (SPN) were proposed because probabilistic performance models allows us to capture the essence of system behaviour through probabilistic assumptions so that a detailed deterministic description of the system can be avoided. The use of exponential distributions for the firing rates of timed transitions is particularly attractive for two reasons. First, exponential timed Petri nets can be mapped onto Continuous Time Markov Chains (CTMC), which lays the foundation for the mathematical solution for SPN (Molloy,1982). Second, the memoryless property of the exponential distribution makes it unnecessary to distinguish between the distribution of the delay itself and that of the remaining delay after a state change. Recognizing that both time-consuming activities and logical behaviours exist in a system, Generalized Stochastic Petri nets(GSPN) allow transitions to be either timed or immediate. In the meanwhile, the state space generated by the GSPN is smaller than that of the SPN for a topologically identical PN model. Some other extended Petri nets are: Coloured Petri nets, which give more compact graphical representation of Petri nets; Extended Stochastic Petri nets, which is an effort to include non-exponential distribution in the analysis of SPN -based models; and DSPN, which are Petri Nets with Deterministic and Exponential Firing times.

GSPN was chosen as the modelling tool because GSPN is the most widely used class of SPN's and is suitable for the problem in question. The package SPNP (Stochastic Petri net Package) developed by (Ciardo,1992) has been used to analyze the GSPN models.

3.4.2 Generalized Stochastic Petri nets

Definition 4: A *generalized stochastic Petri net* is a five tuple

$$GSPN = (P, T, A, M_o, L)$$

where (P,T,A,M_o) is a standard Petri net, and

$L=\{ l_1, l_2, ..., l_m\}$ is the set of exponential firing rates associated with "m" timed transitions.

Definition 5: A Random switch is defined as a set of immediate transitions with relative probability of firing (switching distributions) for resolving conflict when more than one transition is enabled at the same time.

A crucial aspect of the definition of the GSPN is the identification of random switches, and the definition of the correct switching distributions in all markings may sometimes require ingenuity and insight into system operations.

Definition 6: The firing rules of GSPN are as follows:

1) If the set of enabled transition H comprises only timed transitions, then transition t_i, $i \epsilon H$, fires with probability

$$\frac{l_i}{\sum_{k \in H} l_k}$$

2) If H comprises both timed and immediate transitions, immediate ones have priority. If there is only one immediate transition, then this is the one that fires. When H comprises several immediate transitions, they fire according to the switching distribution.

Definition 7: Tangible marking are markings that enable no immediate transitions; otherwise, they are called vanishing markings.

A Generalized Stochastic Petri Net can be mapped into a Markov Chain by removing vanishing markings, since they do not contribute to the measurable behaviour of the model. Therefore the performance analysis by a Petri Net model can be summarized by the Fig. 3.2. Specifically all one has to do is to model the system with a Petri Net. Then, based on the initial marking, the reachability graph can be obtained and analyzed quantitatively.

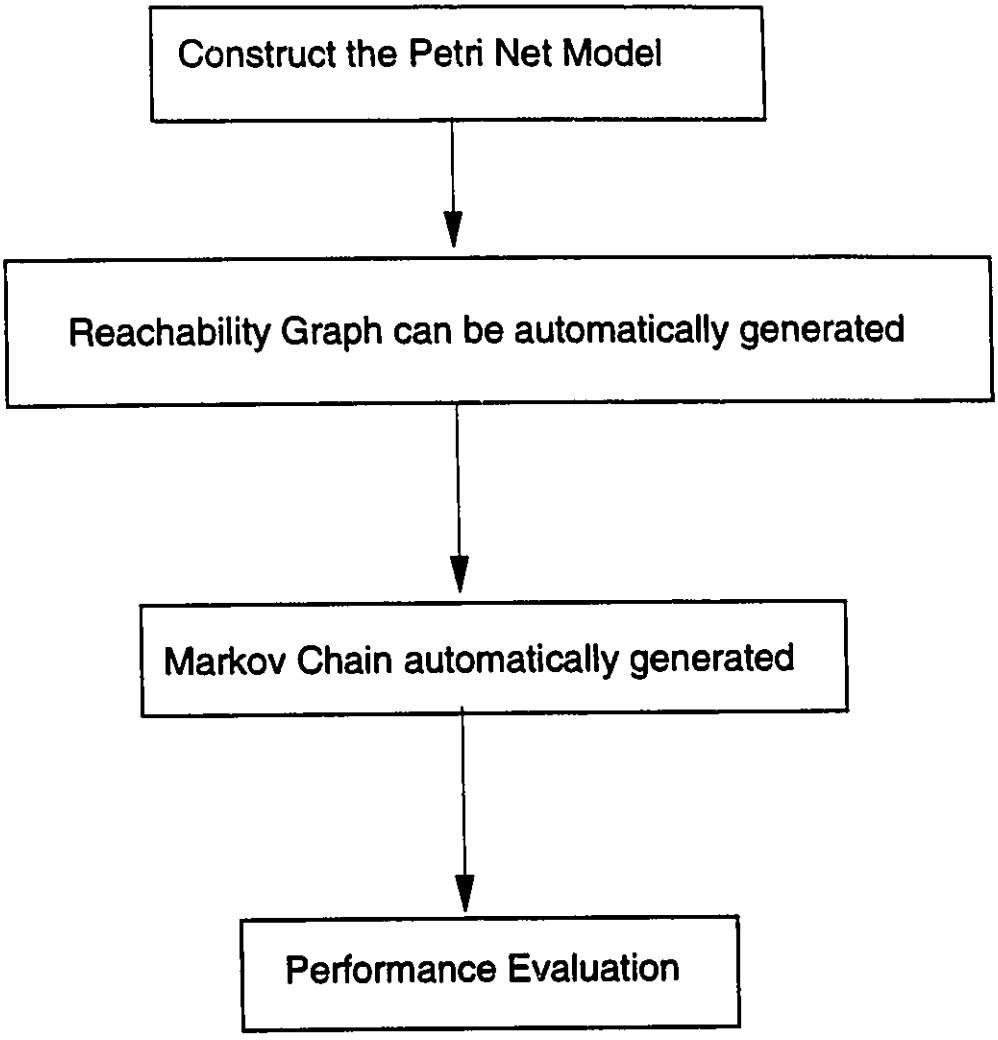


Fig 3.2 Sequence of steps for Petri Net modelling

3.5 Some extensions to Petri nets

Many extensions to the standard petri nets were introduced to increase the modelling power of the tool. Although not all the extensions are used in this research, a list is provided for completeness. The definitions follow:

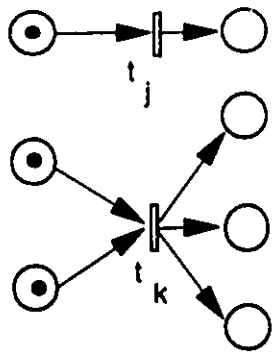
Definition 8: Multiple arcs are arcs associated with multiplicity k, an integer number. An input arc from p to t with multiplicity k requires k tokens in p to enable t, it causes their removal upon firing. An output arc with multiplicity k from t to p causes the addition of k tokens in p when it fires.

Definition 9: An inhibitor arc with multiplicity k from p to t disables t if k or more tokens are in p. In particular, if multiplicity is 1, t is disabled unless p is empty.

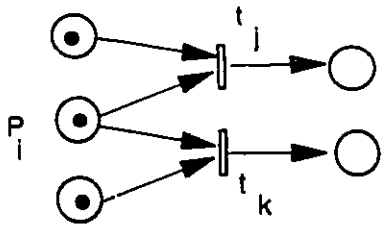
Definition 10: If each transition is assigned a fixed priority (a non-negative integer), the enabling rule is modified so that, in each marking, only the enabled transitions with the highest priority are enabled, while the remaining ones are disabled.

Definition 11: An enabling function E_t can be defined on each transition t. If $E_t(m)=1$, t is enabled in marking m, otherwise t is disabled.

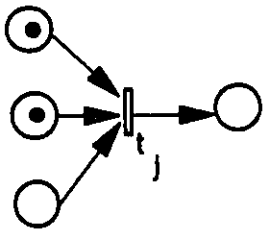
Definition 12: Enabling functions, firing rates, switching distributions, and arc multiplicity can be different in each marking. This is called marking dependence. All these additional structures or the combination of these structures are used to selectively disable a transition in a marking which would otherwise enable it. The following examples Fig. 3.3 shows some of the modelling power of the Petri net models with the aid of the above mentioned extensions (Peterson,1981; Ciardo,1987).



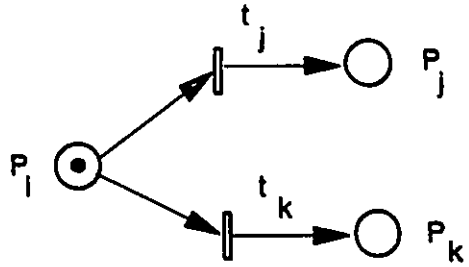
a) Concurrency: The two transitions t_j and t_k can fire in any order.



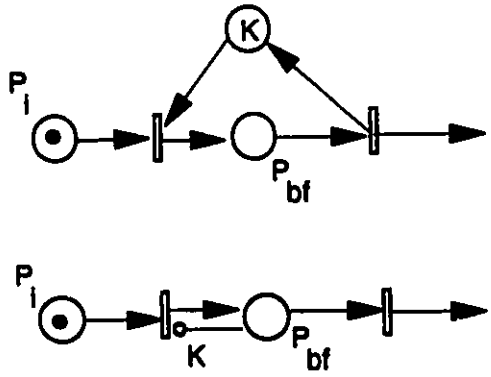
b) Conflict: Transitions t_j and t_k are in conflict since firing either will remove the token from P_i, thus disabling the other transition.



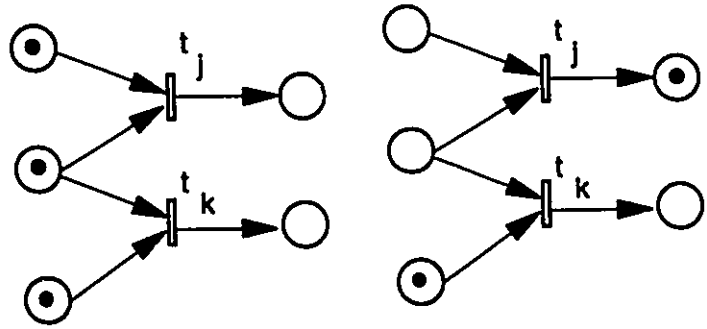
c) Synchronization: Transition t_j will not be enabled until a token arrives at the place currently without a token.



d) Splitting customers: A token in P_i is dispatched to either process t_j or t_k according to the enabling function.



e) Finite buffer sizes and blocking: When $P_{bf} = K$, the token (customer) in P_i is blocked.



f) Priority: By assigning higher priority to t_j other than t_k the conflict is resolved.

Fig. 3.3 Modelling Power of Petri Nets

Chapter 4

ANALYSIS OF MARKOV CHAINS

4.1 Steady State Analysis

The steady state analysis of a manufacturing system can be simply understood by the following example ,(Viswanadham and Narahari,1992).

A manufacturing system comprising a single NC machine that works on an inexhaustible supply of raw workpieces can be modeled as a CTMC. The machine takes an exponentially distributed amount of time to process each part. Each time the machine finishes processing a part, it is set up for the next part, the set up taking an exponentially distributed amount of time. The machine is also prone to failures with time between failures exponentially distributed. The failed machine is immediately repaired, the repair time being exponentially distributed. We have a model with state space {0,1,2} where the states have the following interpretation.

0: Machine being set up for the next part.

1: Machine processing a part.

2: Machine failed, being repaired.

If X(t), for t>=0, represents the state of the system at time t, then {X(t): t>=0} is a

CTMC. Let the rates of the random variables be given by **s=setup rate; p=processing rate; f=failure rate; r=repair rate**. In the above example let us assume that the resume policy is followed which means the whenever there is a machine failure during the processing of a part, work on the same part is continued once the machine is repaired. Fig. 4.1 shows a state transition diagram, which captures the transition among these three states using directed arcs. Each arc is labelled with the corresponding transition rate. The **transition rate matrix** or the **infinitesimal generator Q** of this CTMC model is given by

$$Q = \begin{bmatrix} -s & s & 0 \\ p & -(p+f) & f \\ o & r & -r \end{bmatrix}$$

In state 0, the machine is being set up. After the set up operation, the machine starts processing and so transits to state 1 with rate s. In state 1 there are two possibilities:

(i) machine finishes processing and transits back to state 0, and

(ii) machine fails before finishing processing.

Since the processing time and the time to failure are independent random variables, the transition rate is p in the former case and f in the latter case. In state 2, the machine gets repaired and the next state is always 1 and it is reached with rate r. The entries q_{00}, q_{11}, q_{22} of the Q matrix are computed using the relations

$$q_{00} = -(q_{01} + q_{02})$$

$$q_{11} = -(q_{10} + q_{12})$$

$$q_{22} = -(q_{20} + q_{21})$$

The steady state probability values π_0, π_1 and π_2 of states 0, 1 and 2 respectively are given by

$$\pi_0 s = \pi_1 p$$

$$\pi_1 (p+f) = \pi_0 s + \pi_2 r$$

$$\pi_2 r = \pi_1 f$$

$$\pi_0 + \pi_1 + \pi_2 = 1$$

The first three equations above are the rate balance equations which can be obtained from the state transition diagram. The above equations yield the values

$$\pi_0 = \frac{pr}{pr + rs + fs}; \quad \pi_1 = \frac{rs}{pr + rs + fs}; \quad \pi_2 = \frac{fs}{pr + rs + fs}$$

These are the steady state values of the three states of the system.

4.2 Transient Analysis of CTMC

The dynamical evolution over time of a CTMC can be described in terms of the forward and backward differential equations. For a homogeneous CTMC {X(t):t>=0}, the forward equation and the backward equation are respectively given by,

$$\frac{dH(t)}{dt} = H(t)\ Q$$

and

$$\frac{dH(t)}{dt} = Q\ H(t)$$

where

$$H(t) = [\ p_{ij}\ (t)\ ]$$

with

$$p_{ij}\ (t) = P\ \{\ X(t) = j |\ X(0) = i\ \}$$

Note that Q is the infinitesimal generator of the CTMC and that the initial conditions are given by H(0) = I. Also the above equations expressed in terms of the individual matrix elements are given by

$$\frac{dp_{ij}(t)}{dt} = q_{jj}p_{ij}(t) + \sum_{k \neq j} q_{kj}p_{ik}(t) \qquad (1)$$

$$\frac{dp_{ij}(t)}{dt} = q_{ii}p_{ij}(t) + \sum_{k \neq i} q_{ik}p_{kj}(t) \qquad (2)$$

It is important to note that the forward and backward equations have the same unique solution given by

$$H(t) = \exp(Qt) \qquad (3)$$

If we are interested in computing the state probabilities, that is,

$$\Pi(t) = [\ p_0(t)\ p_1(t)\ p_2(t)\ldots\ldots\ ]$$

where $p_j(t) = P\{X(t) = j\}$, then we need to solve the equation

$$\frac{d\Pi(t)}{dt} = \Pi(t)Q$$

whose solution is given by

$$\Pi(t) = \Pi(0)\ \exp(Qt)$$

The following example will give a feel for the backward and forward equations and their solutions:

Let us look at a single machine system with failure rate = λ; repair rate = μ and two states;

0: Machine working

1: Machine undergoing repair after failing

For this example

$$H(t) = \begin{bmatrix} p_{00}(t) & p_{01}(t) \\ p_{10}(t) & p_{11}(t) \end{bmatrix}; \; Q = \begin{bmatrix} -\lambda & \lambda \\ \mu & -\mu \end{bmatrix}$$

The forward equations obtained from (1) in this case are given by

$$\frac{dp_{00}(t)}{dt} = p_{00}(t)q_{00} + p_{01}(t)q_{10}$$

$$\frac{dp_{01}(t)}{dt} = p_{01}(t)q_{11} + p_{00}(t)q_{01}$$

$$\frac{dp_{10}(t)}{dt} = p_{10}(t)q_{00} + p_{11}(t)q_{10}$$

$$\frac{dp_{11}(t)}{dt} = p_{11}(t)q_{11} + p_{10}(t)q_{01}$$

The backward equations obtained from (2) are given by

$$\frac{dp_{00}(t)}{dt} = q_{00}p_{00}(t) + q_{01}p_{10}(t)$$

$$\frac{dp_{01}(t)}{dt} = q_{01}p_{11}(t) + q_{00}p_{01}(t)$$

$$\frac{dp_{10}(t)}{dt} = q_{10}p_{00}(t) + q_{11}p_{10}(t)$$

$$\frac{dp_{11}(t)}{dt} = q_{11}p_{11}(t) + q_{10}p_{01}(t)$$

First the terms $p_{00}(t)$ and $p_{10}(t)$ are solved using the backward equations:

$$\frac{dp_{00}(t)}{dt} = -\lambda p_{00}(t) + \lambda p_{10}(t) \tag{4}$$

$$\frac{dp_{10}(t)}{dt} = \mu p_{00}(t) - \mu p_{10}(t) \tag{5}$$

Multiplying (4) by μ and (5) by λ and summing, we get

$$\mu \frac{dp_{00}(t)}{dt} + \lambda \frac{dp_{10}(t)}{dt} = 0$$

Integration of the above yields

$$\mu p_{00}(t) + \lambda p_{10}(t) = C$$

where C is a constant. Since $p_{00}(0) = 1$ and $p_{10}(0) = 0$, we obtain $C = \mu$ and hence we have

$$\mu p_{00}(t) + \lambda p_{10}(t) = \mu \tag{6}$$

Using (6) in (4) we get

$$\begin{aligned} \frac{dp_{00}(t)}{dt} &= -\lambda p_{00}(t) + \mu - \mu p_{00}(t) \\ &= \mu - (\lambda+\mu) p_{00}(t) \end{aligned}$$

This gives the differential equation

$$\frac{dp_{00}(t)}{dt} + (\lambda+\mu) p_{00}(t) = \mu \tag{7}$$

Equation (7) is a linear differential equation with solution given by

$$p_{00}(t) = \frac{\mu}{\lambda+\mu} + \left(\frac{\lambda}{\lambda+\mu}\right) e^{-(\lambda+\mu)t} \tag{8}$$

Using (6) in (8) we obtain

$$\begin{aligned} p_{10}(t) &= \frac{\mu}{\lambda}\,[1 - p_{00}(t)] \\ &= \frac{\mu}{\lambda+\mu} - \left(\frac{\mu}{\lambda+\mu}\right) e^{-(\lambda+\mu)t} \end{aligned}$$

Proceeding in a similar way, it can be shown that

$$p_{11}(t) = \frac{\lambda}{\lambda+\mu} + \left(\frac{\mu}{\lambda+\mu}\right) e^{-(\lambda+\mu)t}$$

$$p_{01}(t) = \frac{\lambda}{\lambda+\mu} - \left(\frac{\lambda}{\lambda+\mu}\right) e^{-(\lambda+\mu)t}$$

Fig. 4.2 shows the evolution of the probabilities $p_{ij}(t)$. Note that

$$\lim_{t\to\infty} p_{00}(t) = \lim_{t\to\infty} p_{10}(t) = \frac{\mu}{\lambda+\mu}$$

$$\lim_{t\to\infty} p_{11}(t) = \lim_{t\to\infty} p_{01}(t) = \frac{\lambda}{\lambda+\mu}$$

The above limiting probabilities are precisely the steady state probabilities π_0 and π_1 of the states 0 and 1, respectively.

We shall now compute, for j=0,1

$$\begin{aligned} p_j(t) &= P\{X(t) = j\} \\ &= P\{X(t) = j \mid X(0) = 0\} + P\{X(t) = j \mid X(0) = 1\} \\ &= p_{0j}(t)p_0(0) + p_{1j}(t)p_1(0) \end{aligned}$$

If we assume that the initial state is state 0, then we have $p_0(0) = 1$ and $p_1(0) = 0$. Therefore

$$p_0(t) = \frac{\mu}{\lambda+\mu} + \left(\frac{\lambda}{\lambda+\mu}\right) e^{-(\lambda+\mu)t}$$

$$p_1(t) = \frac{\lambda}{\lambda+\mu} - \left(\frac{\lambda}{\lambda+\mu}\right) e^{-(\lambda+\mu)t}$$

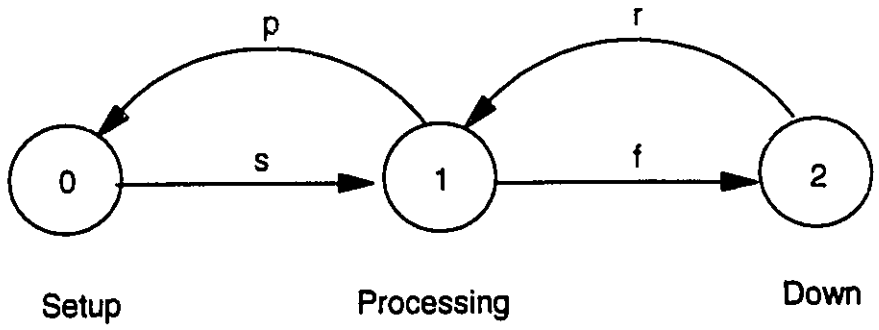


Fig. 4.1 State Transition diagram of the CTMC example

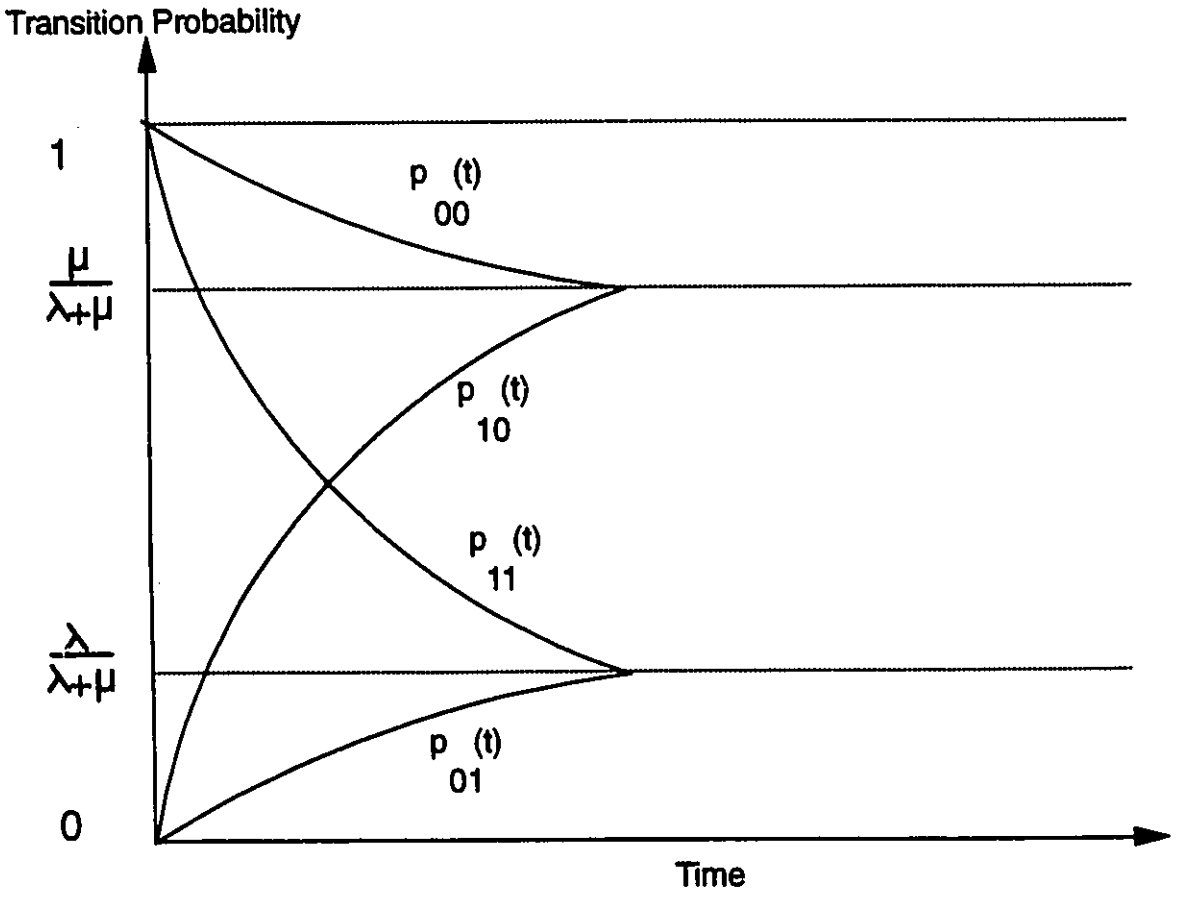


Fig. 4.2 Evolution of transition probabilities

Chapter 5

PETRI NET BASED MODELS OF THE AUCTION BASED HETERARCHICAL MANUFACTURING SYSTEM

5.1 Description of the Petri Net model

The models discussed here have the following common features. Fig. 5.1 gives the schematic representation of the model and Fig. 5.2 gives the Petri Net diagram. Table 5.1 explains the terminologies used to define the Petri Net shown in Fig. 5.2.

There are six machining centres in the system. The machines have processing rates which are **exponentially** distributed. The system has 3 part types. Each part type may have multiple parts within them to be manufactured. Each part type has a process plan. For our model each process plan has been allowed to have a maximum of 6 stages. Alternative machining centres for doing the same work have been allowed for each process stage. The process plan and the process stages within them for each of the 3 part types have been shown in Tables 5.2 to 5.4. The six machining centres have been divided into groups of three. Machines 1,2 and 3 are considered as upstream and machines 4,5 and 6 are considered as downstream machines. A part progresses along its specified process plan by shuttling between the group of upstream and downstream machines

alternatively. For example if Part type 2 is in stage 3 then it has to choose from among upstream machines 1,2 and 3. It chooses the best among these using the heuristic specified, which for example, is the " Shortest Queue Length" as in Model 1. In addition to the machining centres, the model also has a communication subnet that models the features of a token bus protocol based communication network. Bids are sent in chronological order to the alternatives available at any stage. For example, if m/c's 4,5 and 6 are available as alternatives for a particular stage then the bid is first sent to m/c 4, then to m/c 5 and finally to m/c 6. This sequence is important since this sequence decides which machine gets chosen in the case of a tie. This has been elaborated upon in the subsequent sections of the chapter.

There are three identical subnets identifiable from Fig. 5.2. Each subnet represents one part type. Fig. 5.1 represents the flow of logic within each Petri subnet. A part waiting for an upstream machine has to use the heuristic to select the best upstream machine available. At this stage, bid requests are only sent to those upstream machines which figure as available alternatives to this part type for its corresponding stage in the process plan. (Ref. Tables 5.2 to 5.4). All parts leaving upstream machines have to pass through the communication subnet before reaching the next stage of bidding. This is represented by the second heuristic box in front of the downstream machines. Just like in the case of the upstream machines, the best downstream machine is chosen by first selectively sending bids to the allowable alternatives and then choosing the best on the basis of the

heuristic.The parts released from the downstream machines go through the same cycle of machine selection repeatedly till all stages in there process plan are completed. While the features discussed above are common to all the following models, they differ from each other in terms of the tie-breaking rules or part mix or the machining rates of the machining centres. Further details on these differences is given in the following sections.

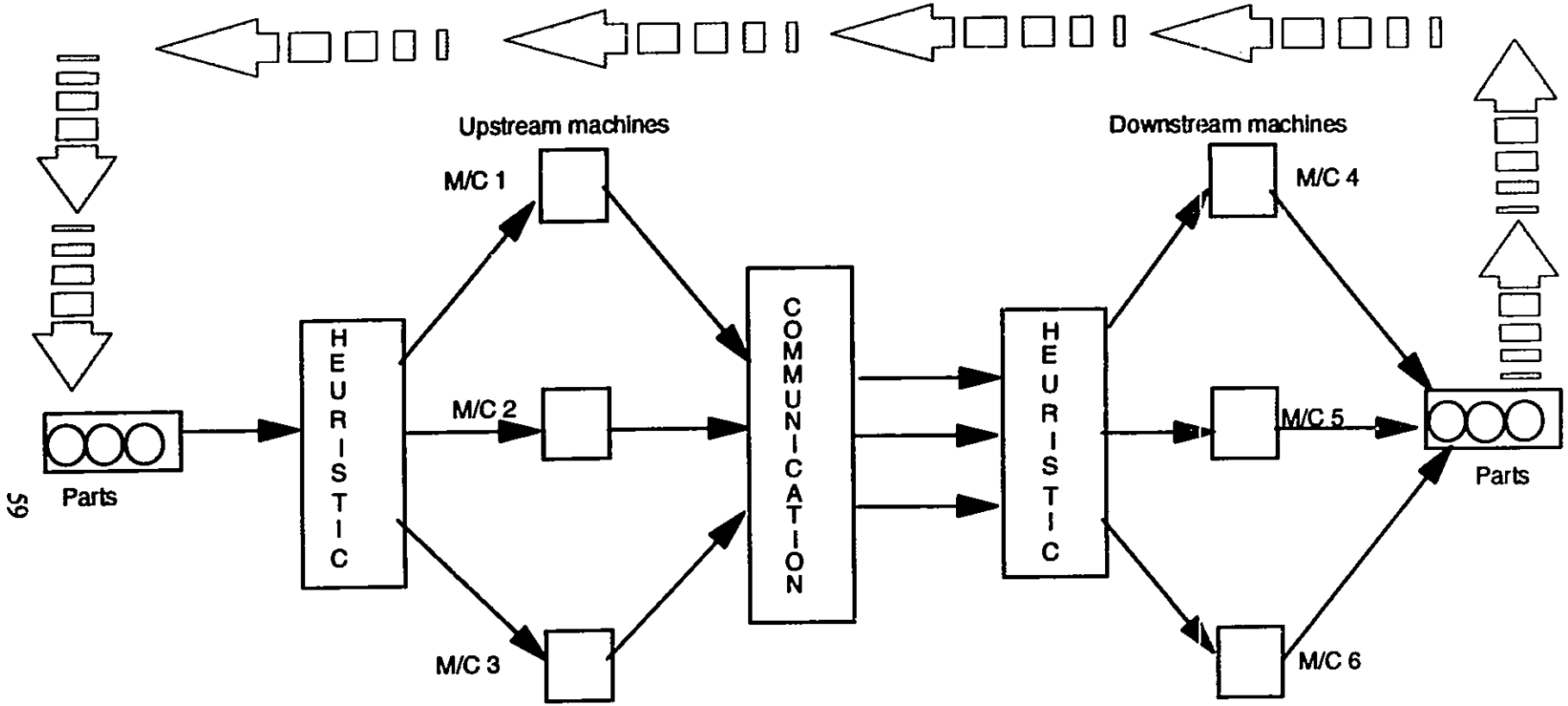


Fig 5.1 Schematic of the Auction Based Manufacturing System

Table 5.1 Terminologies used in Fig 5.2

	Symbol	Physical Meaning
Places	psf,2	Starting point for all three part types
	psfi,0	Upstream machine selection stage for part type i
	pui,n	Upstream machine n=1,2,3 is machining part type i
	pci,0	Counter for determining the stage in which part type i is currently in
	psfi,1	Part type i leaves upstream machine
	pcommi,0	Communication bus is available to part type i for transmission of bid requests
	pcommi,1	Part type i sends out bid requests
	pdi,n	Downstream machine n=4,5,6 is machining part type i
Transitions	tdii,3	Immediate transition, part type i is ready to enter shopfloor
	tuii,n,0	Immediate transition, part type i is about to enter upstream machine n
	tuii,n,1	Timed transition, part type i is being processed at upstream machine n
	tcommi,0	Timed transition, bid request from part type i is about to enter communication subnet
	tcommi,1	Timed transition, transmission delay of bid request from part type i
	tbufsi,n	Immediate transition, Part type i is about to enter downstream machine n
	tdii,n	Timed transition, Part type i is being machined by downstream machine n

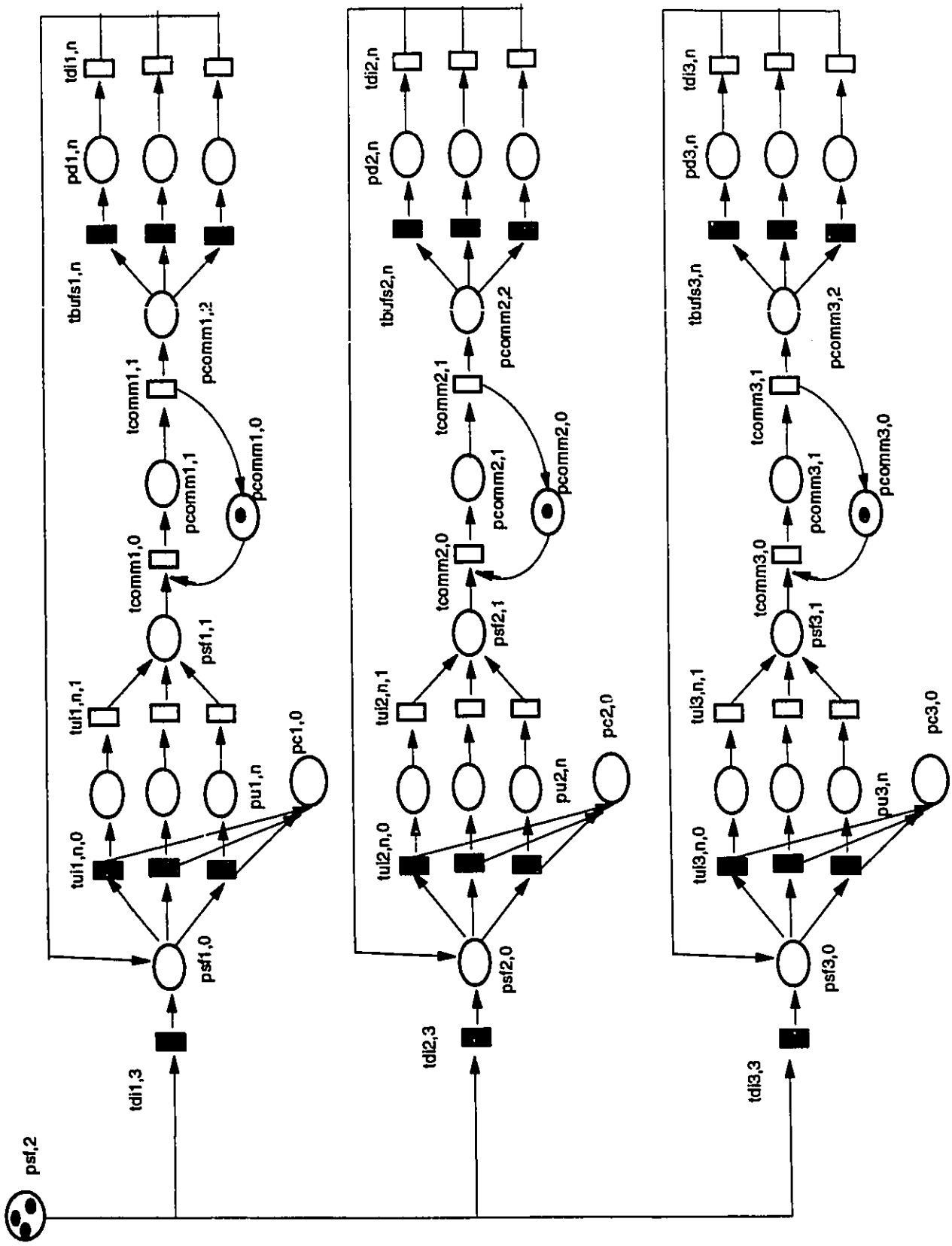


Fig. 5.2 Petri Net representation of the Model

Table 5.2 Process plan for part type 1

Part Type 1	Alternatives available
Stage 1	m/c 2 , m/c 3
Stage 2	m/c 5 , m/c 6
Stage 3	m/c 1 , m/c 2 , m/c 3
Stage 4	m/c 5 , m/c 6
Stage 5	m/c 2 , m/c 3
Stage 6	m/c 5 , m/c 6

Table 5.3 Process plan for part type 2

Part Type 2	Alternatives available
Stage 1	m/c 2 , m/c 3
Stage 2	m/c 5 , m/c 6
Stage 3	m/c 1 , m/c 2
Stage 4	m/c 5 , m/c 6
Stage 5	m/c 1 , m/c 2
Stage 6	m/c 5 , m/c 6

Table 5.4 Process plan for part type 3

Part Type 3	Alternatives available
Stage 1	m/c 3
Stage 2	m/c 4 , m/c 5 , m/c 6
Stage 3	m/c 2 , m/c 3
Stage 4	m/c 5 , m/c 6
Stage 5	m/c 1 , m/c 2
Stage 6	m/c 5 , m/c 6

5.2 Specific versions of the model

5.2.1 Model I (SQL TFIRST 5,26,13)

This model is based on the "Shortest Queue Length" with TFIRST rule. At each stage the best machine is chosen from among the alternatives available in favour of the machine which has the least queue length or the least number of parts waiting to be processed. In case there is a tie between machines, then the first one among the alternatives is selected. The number of parts within each of Part Types 1, 2 and 3 are 5,26 and 13 respectively.

5.2.2 Model II (SQL TLAST 5,26,13)

This model is based on the "Shortest Queue Length" with the TLAST rule. In the case of a tie between the different alternatives available to machine a stage the last among the machines is selected.

5.2.3 Model III (SQL TFIRST HSPEED 5,26,13)

This model is based on the Shortest Queue Length rule but differs from Model I in terms of the machining rates of Downstream m/c's 5 and 6 for machining part type 3. In this case these rates are higher, hence this case is called the HSPEED case. The number of parts of each part type are however the same as in Model I.

5.2.4 Model IV (SQL TFIRST 50,26,13)

This model is based on the Shortest Queue Length rule but differs from Model I in the number of parts of Type 1 being handled. Unlike Model I this model handles 50 parts of Type 1, 26 parts of Type 2 and 13 parts of Type 3.

5.2.5 Additional features of the model

The models that have been developed have the following additional features over the previous prototype developed by Zhou (1993).

1) The concept of process plans was introduced to the model. The model now has a unique process plan for each part type in the system. The number of stages in each process plan has been taken to be 6. Alternatives at each stage of the 6 stages in a process plan have been provided. The new model can also accommodate larger number of process stages within each process plan by the addition of a few more enabling functions to regulate token flow as per the newly added process stages. Of course, the state space will increase in size and must be computationally feasible. Any part type with a lower number of process stages can also be accommodated in this model, since, for such a part type the enabling function can be suitably modified so that token flow is stopped within its own subnet as soon as the requisite number of process stages have been covered.

2) The token flow is regulated based on marking dependant enabling functions which represent the process stages of each part type. During the state space generation, the system is aware of the stage in which each part type resides in at any particular instance of time with the help of a "counter". The "counter" is actually a place in which tokens keep accumulating every time the part chooses a new machine (either upstream or downstream) for the next stage of the machining in the process plan. The marking dependence of the enabling functions have been tied into this "counter' place which helps the enabling function to regulate the token flow strictly as per the process plan.

3) While the process plans provide a rough guideline as to the sequence in which a part will be manufactured, it does not specify which machines the part should go to exactly. This ambiguity is because of the alternatives provided in the process plan of each part type at each of its machining stages. This ambiguity is actually resolved at the time of the state space generation with the help of "heuristic" and tie breaking rules which also represent the control approach for the machining routine. For example this heuristic could be the Shortest Queue Length or the Least Waiting Time and the tie breaking rule could be TFIRST or TLAST. Any combination of these rules can be incorporated into the enabling functions to regulate the token flow accordingly.

Auction based manufacturing is a type of manufacturing that gives little idea about how parts will actually be allocated to machines during the actual machining process or how machines will be utilized and what the part mix will be for each machine in the FMS. This so called "uncertainty" arises because of one of the biggest strengths of auction based manufacturing which is the use of intelligent machines which result in the real-time evolution of schedules based on the prevailing system status. It would be of tremendous help to an operator if there was a way by which it could be known in advance (inspite of all the uncertainty caused by real-time schedule generation) as to how the system is "most likely" to behave. These models developed using Petri Nets address exactly this issue They predict the most likely behaviour of the system and allow the operator to explore alternative control algorithms to improve part flow within the system or improve machine utilization or throughput.

5.3 Assumptions made in developing the model

1) All the parts of a particular part type are assumed to be grouped together as one job. This implies that whenever a machine takes up the machining of one part type, it completes machining all the parts in that part type before moving on to the next part type awaiting service.

2) The buffer of a machine can hold all the parts that arrive for machining. The buffer capacity for each machine is not a constraint.

3) A part has to go to an upstream machine and a downstream machine alternately as it goes along its prescribed process plan.

4) The alternatives available for a particular stage all belong to the same group, that is, they are all either upstream machines or downstream machines.

5) The availability of transmission capacity and AGV's for transporting parts is a assumed to be sufficient and doesn't act as a constraint in this study.

6) Machine breakdowns have not been taken into consideration.

7) Tools are available to each machining centre whenever they are required.

5.4 Parameters used to analyse and evaluate each of the models

5.4.1 Parameters derived from the steady state analysis:

1) The Utilization level of each machine

For upstream machine j for part type 1, machine utilization, UTU1(j), is the probability that machine j is busy. In terms of the Petri Net model, it is the probability that transition (tui1,j,1) is enabled, i.e.,

$$UTU1(j) = Pr(tui1,j,1 \ enabled) \quad \forall j \tag{9}$$

Similarly for the downstream machine j of say part type 3 the Utilization is calculated using

$$UTD3(j) = Pr(tdi3,j \text{ enabled}) \quad \forall\, j \tag{10}$$

These expressions will give the probability of finding at least one token in the corresponding place in the Petri Net. For the first case the corresponding place will be **pui1,j** while in the second case it will be **pdi3,j**. In other words, this probability value is a representation of the percentage amount of time that a machine will be found to be busy during the machining cycle. Hence this parameter actually represents the utilization of a machine. It should be noted that the utilization value of a machine may decrease when it is speeded up or when it takes up parts requiring lesser amount of machining time. This is because utilization is a relative term calculated with respect to the total time for completing all parts in the system, hence if a machine finishes up its job faster, the ratio of the time for which it is found to be busy reduces and hence utilization is found to reduce.

2) The Average throughput of each machine

This is calculated using the following equation for the corresponding transition (Ciardo,1992).

The average throughput of transition θ is

$$TH(t_\theta) = \sum_{i \in R(\theta)} p(M_i) * \mu(\theta, M_i) \tag{11}$$

where $R(\theta)$ is the subset of reachable markings that enable transition θ, $p(M_i)$ is the

probability of Marking M_i, and $\mu(\theta,M_i)$ is the rate of transition θ in marking M_i. This equation yields the rate at which parts will flow through the transition, or in other words this is the rate at which parts will be flowing out of the corresponding machine. For an exponential transition with non-marking dependant firing rate, the average throughput is the product of the probability that the transition is enabled, $p(\theta)$, and its firing rate, $\mu\theta$. According to the definition of utilization, we have

$$TH(t_\theta) = p(\theta)*\mu(\theta) = UT(\theta)*\mu(\theta) \tag{12}$$

For upstream machine j for part type 1, the throughput, THU1(j), is the throughput of the transition tui1,j,1, i.e.,

$$THU1(j) = throughput(tui1,j,1) \quad \forall\, j \tag{13}$$

For downstream machine j in the case of part type 2, the throughput will similarly be given by

$$THD2(j) = throughput(tdi2,j) \quad \forall\, j \tag{14}$$

The average throughput of an upstream machine j considering all three part types will be given by

$$THU(j) = throughput(tui1,j,1) + throughput(tui2,j,1) + throughput(tui3,j,1) \quad \forall\, j \tag{15}$$

5.4.2 Parameters derived from the transient analysis:

1)The expected value of the number of parts

The expected value of the number of parts of part type 1 at a machine j at time t is given by

$$Exp(j,1,t) = \sum_{P_i \ni P(t)} p_{jl}(t) * N_{jl}(t) \qquad (16)$$

where P(t) is the set of all states with finite transition probability values at timepoint t, $p_{jl}(t)$ is the transition probability of a state in which machine j is busy with part type 1 at time t and $N_{jl}(t)$ is the number of parts in that state that belong to part type 1 and are at machine j at time t.It may be noted that the estimation of the transition probability is difficult and cumbersome for the model under consideration because of the very large state space (13000 states). The equations representing the state transition probabilities would be extremely large and cumbersome if calculated using the basics of Queuing Theory i.e. using Eqns (1) and (2). One can get an idea of the complexity of the transition probability equations from Eqns (4) and (5) which have been calculated for a very simple case with only 3 states. Through use of the Petri Net software, the derivation of these transition probabilities has been automated. In the present case therefore, $P_{jl}(t)$ has been automatically derived by the Petri Net software using Eqns (1) and (2).

The expected value of the total number of parts at machine j at time t is given by

$$Exp(j,t) = Exp(j,1,t) + Exp(j,2,t) + Exp(j,3,t) \quad \forall\, j \tag{17}$$

5.5 Decision making process at the enabling function

The decision to select or reject a prospective machine depends on the calculation of the decision parameter. This parameter reflects the control heuristic being used for controlling the parts allocation. In our case, this is the Shortest Queue Length (SQL) rule. If a part has an option of selecting any one from m/c's 4,5 and 6, then the decision parameter for each of them will be calculated by the expression,

$$DP(j,t) = \sum N_{j1}(t) + N_{j2}(t) + N_{j3}(t) \tag{18}$$

where DP(j,t) is the decision parameter for machine j calculated based on the system status at time t and $N_{j1}(t)$ is the number of parts of type 1 at machine j at time t.

In the case of the example considered above the criteria for part allocation will be

$$j = Min\{\, DP(j,t) \,\} \quad for\ j = 4,5,6 \tag{19}$$

since the SQL heuristic is used.

Chapter 6

RESULTS AND OBSERVATIONS

6.1 Machine wise loading analysis

6.1.1 Comparison of Model I and Model II

Refer to Tables 6.1 and 6.2.

M/C 1

Refer to Figs. 6.1 and 6.7.

In the TFIRST case m/c 1 has higher throughput and utilization values. This is due to the TFIRST tie-breaking rule which treats m/c 1 preferentially over m/c's 2 and 3 whenever a part type has to choose an upstream m/c.

M/C 2

Refer to Figs. 6.2 and 6.8.

On comparing the loading pattern of part type 1 on m/c 2 in the TLAST and the TFIRST cases, it becomes clear that m/c 2 will be much more instrumental for machining of this part type in the TFIRST case than in the TLAST case. In the TFIRST case it is clear that the parts of this part type visit this machine at least on 2 different occassions whereas in the TLAST case they may at most visit the machine only once.

Also m/c 2 will be finished with part type 1 by time point 150 in the TLAST case, whereas it stands a chance of being busy with the same part type till time 350 in the TFIRST case. This clearly shows a greater importance that m/c 2 gets in machining part type 1 in the TFIRST case and hence shows how changing the control heuristics can affect the utilization level of a particular machine in fabricating a particular part type. Also the difference in times at which part type 2 peaks in each case may be noted. Perusal of Figs. 6.2 and 6.8 along with the process plan of part type 2 from Table 5.2 indicates that it is most likely that in the TLAST case m/c 2 will be chosen for machining stages 3 and 5 while for the TFIRST case the same machine will be chosen for stage 1 only.

Hence unlike part type 1, it is observed that in the case of part type 2 it is the TLAST rule that increases the utilization of m/c 2. From Fig. 6.8 it can be predicted that m/c 2 will be most likely working on stage 3 of part type 2 at around time point 75.0.

M/C 3

Refer to Figs. 6.3 and 6.9.

The graphs of all three part types indicate that m/c 3 is much more in demand in the TLAST case than in the TFIRST case. If it is found that m/c 3 has low maintenance and running costs and comparably good machining quality over the other alternatives, then it would be appropriate to promote the utilization of m/c 3. In such a case TLAST rule would be a good control heuristic to improve the probability of the parts choosing m/c

3 over m/c's 1 and 2. Of course, the primary crirteria for choosing machines will still remain to be the shortest queue length criteria. But whenever a tie situation arises with m/c 3 being one of the alternatives, TLAST will encourage bid allocation to m/c 3.

M/C 4

Refer to Figs. 6.4 and 6.10.

Use of m/c 4 is allowed as an alternative only in stage 2 of part type 3. See Table 5.3. Since in the chronology of bidding m/c 4 is the first one to be considered, hence the utilization of this machine improves in the TFIRST case as compared to the TLAST case.

M/C 5

On comparing Figs. 6.5 and 6.11 one can see a significant variation in the pattern of loading of m/c 5. In the case of part type 1 it is evident that m/c 5 is utilized at least for 2 different stages in the TFIRST case, while in the TLAST case it is in all probability going to be used only once. M/c 5 is better utilized for part types 1 and 3 in the TFIRST case. Now if m/c 5 happened to be an expensive and high accuracy machine that should not be left idle, then it would be appropriate to use the TFIRST rule to improve its utilization. This should lead to a better utilization of available shop floor resources as well as better quality of products.

M/C 6

It becomes clear on comparing Figs. 6.6 and 6.12 that m/c 6 is much better utilized for

machining all three part types in the TLAST case as compared to the TFIRST case. Hence, one might intuitively opt for the TLAST case if the prime objective was to utilize m/c 6 to the best possible extent. However, it should be noted that in the TLAST case m/c 6 is used for stages 2 and 6 for part type 1 (Ref. Fig. 6.46) while in comparison, it is used only for stage 4 in the TFIRST case (Ref. Fig. 6.43). Now, if it is felt that m/c 6 is vital for machining stage 4 then TFIRST may be used, but that would also mean compromising with its overall utilization.

6.1.2 Comparison of Model I and Model III

Refer to Tables 6.1 and 6.3.

M/C 1

Refer to Figs. 6.1 and 6.13.

It should be noted that only the processing rates of m/c's 5 and 6 have been speeded up and this change has only been made for parts of part type 3. See Tables 6.1 and 6.3. All other machining rates have been kept as constant. Since the speeding up of the machines does not affect the bid allocation criteria (which is given by Eqns. 18 and 19) hence there is no significant difference in the loading patterns of part types 1 and 2. However, the speeding up of downstream m/c 's does affect the loading of upstream m/c 1 in the case of part type 3. This manifests itself as increased loading and consequently higher utilization and the throughput of m/c 1.

M/C 2

Refer to Figs. 6.2 and 6.14.

Similar to the case of m/c 1, it is found that there is no significant change in the loading pattern of part types 1 and 2. However, there is an increase in the loading of part type 3. This can be attributed to the speeding up of the downstream processes for part type 3, thus resulting in a greater rush of parts onto the upstream side.

M/C 3

Refer to Figs. 6.3 and 6.15.

M/C 3 shows a slight increase in the loading pattern of part type 3. This can be attributed to the speeding up of the downstream m/c's for machining this part type.

M/C 4

Refer to Figs. 6.4 and 6.16.

M/C 4 is a unique machine available as an alternative only to part type 3 and specifically, only for its second stage of machining. This machine is naturally chosen over m/c's 5 and 6 in both the TFIRST and the TLAST cases because of the following two reasons:

1) SQL gives a higher priority to the first machine among the alternatives, hence m/c 4 has a natural advantage over m/c's 5 and 6.

2) M/C's 5 and 6 are much more sought after as compared to m/c 4. Hence there is a high probability that m/c's 5 and 6 will be busy while m/c 4 is found to be idle at the

same time. Hence m/c 4 gets chosen logically.

Because of these two reasons it would be reasonable to conclude that m/c 4 gets chosen whenever the option of choosing it arises. As a result speeding up of m/c's 5 and 6 does not affect the loading pattern of m/c 4. Moreover it ought to be remembered that m/c 4 itself was not speeded up. Hence there is no variation in its loading pattern.

M/C 5

Refer to Figs. 6.5 and 6.17.

There is a significant improvement in the throughput of m/c 5 for part type 1 in the HSPEED case. This conforms to the logical trend expected since throughput ought to increase if a machine is speeded up. A decline is seen in the utilization value in the case of part type 3. This is because m/c 5 has been speeded up. By increasing the pace at which m/c 5 finishes part type 3, the total time taken by m/c 5 to finish all parts of part type 3 reduces. Hence utilization of m/c 5 for the case of part type 3 (which is nothing but the percentage amount of time spent on machining part type 3) actually reduces.

M/C 6

Refer to Figs. 6.6 and 6.18.

The overall utilization level of m/c 6 is found to decline in the HSPEED case. It should be kept in mind that m/c 6 as such is not a favoured machine since the TFIRST rule always prefers m/c 5 over m/c 6 whenever there is a tie. Hence m/c 6 is very susceptible to further changes that promote the usage of m/c 5. By speeding up m/c 5 more jobs get

drawn into m/c 5, especially in the case of part type 1, hence the loading of part type 1 onto m/c 6 reduces. The decline in the loading of m/c 6 with part type 3 may be attributed to the same reason as mentioned earlier for the case of m/c 5.

6.1.3 Comparison of Model I and Model IV

Refer to Tables 6.1 and 6.4.

M/C 1

Refer to Figs. 6.1 and 6.19.

By increasing the number of parts of part type 1 from 5 to 50, it would be natural to expect machines involved in machining part type 1 to take a longer amount of time to finish with these parts. This can be observed from Figs. 6.1 and 6.19 especially from the graphs of part type 1. There is no significant variation in the loading pattern of part types 2 and 3 since the number of parts involved does not differ in the two studies.

M/C 2

Refer to Figs. 6.2 and 6.20.

As expected there is an improvement in the utilization of m/c 2 for part type 1 in the 50,26,13 case. The decline in the utilization of m/c 2 for part type 2 doesn't necessarily indicate reduction in the number of cases of part type 2 coming to m/c 2. Instead, it is due to the fact that the total busy time of m/c 2 has increased and the fraction of time used for machining parts of part type 2 has decreased.

M/C 3

Refer to Figs. 6.3 and 6.21.

The utilization of m/c 3 for part type 2 improves in the 50,26,13 case. This indicates that with more parts keeping m/c 2 busy, the SQL rule diverts parts to the next best alternative, which is m/c 3.

M/C 4

Refer to Figs. 6.4 and 6.22.

The loading pattern of m/c 4 is unaffected by the increase in the number of parts of part type 1. This is in tune with the logical behaviour expected since m/c 4 is only allowed as an alternative in the process plan of part type 3. Hence only an increase in the number of parts of type 3 should affect its loading.

M/C 5

Refer to Figs. 6.5 and 6.23.

M/C 5 shows an increase in the time span that it spends on part type 1. This is in tune with the expected trend.

M/C 6

Refer to Figs. 6.6 and 6.24.

M/C 6 shows an increase in the time span it spends on part type 1. The decline in the utilization of m/c 6 for machining parts of part type 3 could be due to the fact that the number of parts of this type that arrive for machining at m/c 6 remain constant even

though the total use span of m/c 6 increases.

6.2 Part wise loading analysis

6.2.1 Comparison of Model I and Model II

Part Type 1

Refer to Figs. 6.25, 6.26 and Figs. 6.43, 6.46.

Throughput of m/c 1 is more in the TFIRST than in the TLAST case. This is because of stage 3 where m/c 1 gets chosen in the TFIRST case. Since the process plan has m/c 1 at only one stage, it is fairly easy to predict the time at which there is the highest likelihood of it being used for machining stage 3 of part type 1. It is approximately time-point 50.0.

Utilization and Throughput values for m/c 2 are more for the TFIRST case than the TLAST case. This is logical since in the TFIRST case m/c 2 is chosen over other alternatives at stages 1 and 5 whenever there is a tie. From the TLAST case it is evident that m/c 2 is most likely going to be used for machining stage 1 only and will not be used for Type I machining after time point 175. On the contrary, from the TFIRST case it is clear that m/c 2 gets loaded at least twice (i.e. it does at least two stages of Type I). Going by the pattern of the peaks in the TFIRST case, it can be predicted that m/c 2 will be used for stages 1 & 5 while m/c 1 will be used for stage 3. Going by the same reasoning, it can be predicted in the TLAST case that m/c 3 will be used for stages 1 &

5 (See pattern of the peaks of m/c 3 relative to that of m/c 1 in Fig. 6.46) while m/c 1 will only be used for stage 3 of the machining. There is also a likelihood of m/c 3 being used for stages 1 & 3 in the TFIRST case but there is no evidence of it being used for stage 5. This information can be used for optimizing resource allocations.

M/C 4 is not used at all in either case since it doesn't figure as an alternative in the process plan of Type I.

In the TFIRST case m/c 5 is most likely going to be used twice for stages 2 and 6 of the process plan. In comparison m/c 6 is only going to be used for stage 4 in the TFIRST case.

The loading of m/cs 5 and 6 are going to be reversed in the TLAST case. M/c 5 is most likely going to be used for stage 4 while m/c 6 is going to be chosen for stages 2 and 6.

Part Type 2

Refer to Figs. 6.27, 6.28 and Figs. 6.44, 6.47.

Going by the graphs, it can predicted that in the TFIRST case there is a high likelihood of m/c 1 being used for machining of stage 3. It can also be predicted that this m/c 1 will most probably be used in the time frame 75-125. In comparison, there is a much lesser probability that m/c 1 will be used for the SQL TLAST case.

It is noticed that m/c 2 has a higher Utilization and Throughput value in the TLAST case as compared to the TFIRST case. This is because m/c 2 gets predominantly chosen

over m/c 1 when tying with it for machining stages 3 and 5. For stage 1 of the TFIRST case it appears that m/c 2 is preferred over m/c 3. In the TLAST case m/c 2 is still preferred, but there is a marginal improvement in the use of m/c 3.

M/C 3 has a higher utilization and throughput value in the TLAST case since it is preferred over m/c 2 in case of a tie at stage 1. M/c 3 if used will be used approximately at time point 25 for stage 1 of part type 2.

M/C 4 is not used at all for machining Part Type II. Hence it has Utilization and Throughput values of 0.

M/C 5 has a higher Utilization and Throughput value for the TLAST case. That indicates that there are more states with a busy m/c 5 in the TLAST case. Hence, it implies that there is greater likelihood of finding m/c 5 busy with part type 2 in the TLAST case. It can similarly be concluded that there is a greater likelihood of m/c 6 being busy with part type 2 in the TLAST case.

From the patterns of the graphs for m/cs 5 and 6 in the TFIRST case, it can be predicted that the most likely choice for stages 2, 4 and 6 in the TFIRST case will be m/c 5, m/c 6 and again m/c 5. The choices for the same stages in the TLAST case is most likely to be m/c 5, m/c 6 and again m/c 6.

Part Type 3

Refer to Figs. 6.29, 6.30 and Figs. 6.45, 6.48.

M/C 4 has a high Utilization and Throughput value in the TFIRST case. That is because it is chosen over machines 5 and 6. It is most likely going to be used at approximately time point 25. There is no probability of the same machine being used for machining Part Type III in the TLAST case.

M/C 5 has a high Utilization and Throughput value in the TFIRST case. It is logically in tune with what is expected since in the TFIRST case m/c 5 is preferred over m/c 6 whenever there is a tie.

M/C 6 has a higher Utilization and Throughput value in the TLAST case. It is true for the same reason as given above. Either of m/c 5 or m/c 6 will be used for stage 2 of the TLAST case, but m/c 4 will definitely not be used. Hence m/c 4 need not be included as an alternative in the process plan at all. Thus the resources associated with running m/c 4 can be reallocated and used elsewhere.

6.2.2 Comparison of Model I and Model III

Part Type 1

Refer to Figs. 6.31, 6.32 and Figs. 6.43, 6.49.

The Utilization and Throughput values of m/c's 1,2 and 3 do not change significantly with change in the speed of downstream machines for machining part type 3. Among the downstream machines choice of m/c 5 dominates over choice of machine 6 because of the TFIRST rule. Speeding of both m/c 5 and 6 leads to parts coming to the

downstream side faster. Since most of these parts are channelized towards m/c 5 hence utilization and throughput of m/c 5 is seen to improve while the utilization and the throughput values of m/c 6 are found to decline.

Part Type 2

Refer to Figs. 6.34,6.35 and Figs. 6.44,6.50.

Again as seen in the case of part type 1 there is little variation in the performances of m/c 1,2 and 3 in terms of there loading with part type 2.

M/C 4 is not used at all, hence it does not show in the graph. The Utilization and the Throughput values of m/c 5 improves because m/c 5 is the preferred m/c as compared to m/c 6. Hence it takes up the onrush of new parts coming to the downstream side and hence we see an improvement in its Utilization and Throughput values.

Part Type 3

Refer to Figs. 6.35, 6.36 and Figs. 6.45, 6.51.

For this part type, the speeding up of the downstream machines should lead to more number of parts stacking up on the upstream side. This can be seen quite clearly in the cases of m/c's 1,2 and 3 whose throughput values are almost double in the speeded up case as compared to the normal case. The steady state probability values determine the utilization and the throughput, hence a change in machine rates affects the throughput

and the utilization. The usage pattern of m/c 4 does not change even though the total time for which part type 3 is likely to be in the system drastically reduces from 250 time units to 75 time units. This logically should lead to an improvement in the utilization of m/c 4. This is observed from the steady state analysis results in Table 6.3. The speeding of m/c's 5 and 6 lead to a fall in their utilization because now they can finish the parts of type 3 coming to them at a faster rate; hence they are idle(with respect to part type 3) for longer periods of time.

Overall, the Utilization of m/c's 2,3 & 4 are seen to be improving significantly while the utilization values of m/c's 5 & 6 are found to deteriorate. Also, the performance in terms of utilization values of m/c 1 is found to have improved marginally.

6.2.3 Comparison of Model I and Model IV models

Part Type 2

The Figs. 6.44 and 6.53 and the Tables 6.1 and 6.4 indicate an improvement in the utilization and the throughput values of m/c 1. There is also a decline in the utilization and throughput values of m/c 2. These two observations taken together can be explained as follows. Since there are more parts being handled in the system and since the parts prefer m/c 1 to m/c 2, hence the utilization of m/c 1 improves with an increase in the number of parts in the system. For the same reason the same parameters for m/c 2 decrease in value. The Utilization and Throughput values of m/c's 5 & 6 are found to

remain fairly constant through this exercise.

Part Type 3

Refer to Figs. 6.45 and 6.54 and Tables 6.1 and 6.4.

The throughput and the utilization values of m/c's 1,2 and 3 for machining type 3 do not vary much with additional parts of type 1. There is a significant increase in the utilization and the throughput values of m/c 5 while there is a decline in the value of the same parameter for m/c 6. The utilization and the throughput of m/c 4 does not change significantly. This is because almost all the parts of type 3 come to it for machining stage 2 and any increase in the part type 1 does not affect in any way the number of parts of type 3 coming to m/c 4 since m/c 4 is used exclusively in the machining of part type 3 alone.

Table 6.1 Performance evaluation parameters calculated from Model I

SQL TFIRST RULE 5,26,13		M/C 1	M/C 2	M/C 3	M/C 4	M/C 5	M/C 6
Part Type 1	Throughput	0.009	0.03	0.03	0	0.0497	0.0137
	Utilization	0.0708	0.1199	0.2424	0	0.3974	0.1097
	Peak expected number of parts	0.8246	1.6779	2.05	0	1.9552	1.2322
	M/C Rate (units/time)	0.025	0.05	0.025	0.05	0.025	0.025
Part Type 2	Throughput	0.2808	0.1672	0.0292	0	0.2390	0.1157
	Utilization	0.5422	0.3214	0.0374	0	0.023	0.0278
	Peak expected number of parts	12.498	18.465	3.406	0	0.869	1.828
	M/C Rate (units/time)	0.02	0.02	0.03	0.2	0.4	0.16
Part Type 3	Throughput	0.1166	0.2974	0.3770	0.5274	0.1564	0.1073
	Utilization	0.010	0.0254	0.0322	0.4057	0.301	0.165
	Peak expected number of parts	0.2459	0.1991	0.8173	7.1734	7.1176	2.5019
	M/C Rate (units/time)	0.9	0.9	0.9	0.1	0.04	0.05

Table 6.2 Performance evaluation parameters calculated from Model II

SQL TLAST RULE 5,26,13		M/C 1	M/C 2	M/C 3	M/C 4	M/C 5	M/C 6
Part Type 1	Throughput	0.0039	0.0068	0.0525	0	0.0435	0.0255
	Utilization	0.0314	0.027	0.4218	0	0.3492	0.2039
	Peak expected number of parts	0.2248	1.678	2.1356	0	1.8916	1.6719
	M/C Rate (units/time)	0.025	0.05	0.025	0.05	0.025	0.025
Part Type 2	Throughput	0.0198	0.3973	0.0881	0	0.3898	0.2395
	Utilization	0.0380	0.7641	0.1131	0	0.0375	0.0576
	Peak expected number of parts	2.7711	18.0888	6.8112	0	1.1026	1.3696
	M/C Rate (units/time)	0.02	0.02	0.03	0.2	0.4	0.16
Part Type 3	Throughput	0.1507	0.0704	0.3527	0.0086	0.0798	0.4849
	Utilization	0.0129	0.0060	0.0301	0.00663	0.1536	0.7466
	Peak expected number of parts	0.1826	0.1741	0.8172	0.013	1.3435	10.7277
	M/C Rate (units/time)	0.9	0.9	0.9	0.1	0.04	0.05

Table 6.3 Performance evaluation parameters calculated from Model III

SQL TFIRST RULE, HSPEED 5,26,13		M/C 1	M/C 2	M/C 3	M/C 4	M/C 5	M/C 6
Part Type 1	Throughput	0.0091	0.0288	0.031	0	0.0627	0.0062
	Utilization	0.0726	0.115	0.248	0	0.501	0.0492
	Peak expected number of parts	0.8393	1.678	2.052	0	2.3997	0.2808
	M/C Rate (units/time)	0.025	0.05	0.025	0.05	0.025	0.025
Part Type 2	Throughput	0.2756	0.1710	0.0312	0	0.2405	0.2384
	Utilization	0.532	0.3289	0.0400	0	0.0231	0.0573
	Peak expected number of parts	12.2794	18.5286	3.4056	0	0.944	1.3085
	M/C Rate (units/time)	0.02	0.02	0.03	0.2	0.4	0.16
Part Type 3	Throughput	0.2127	0.5265	0.6734	0.9412	0.2444	0.2262
	Utilization	0.0182	0.045	0.0575	0.7244	0.0268	0.0194
	Peak expected number of parts	0.7051	0.2487	0.8173	7.1734	1.3517	0.3364
	M/C Rate (units/time)	0.9	0.9	0.9	0.1	0.7	0.9

Table 6.4 Performance evaluation parameters calculated from Model IV

SQL TFIRST RULE, 50,26,13		M/C 1	M/C 2	M/C 3	M/C 4	M/C 5	M/C 6
Part Type 1	Throughput	0.0093	0.0308	0.03	0	0.0398	0.0305
	Utilization	0.0743	0.1231	0.2401	0	0.318	0.2431
	Peak expected number of parts	7.6657	23.9741	24.4592	0	20.675	5.7125
	M/C Rate (units/time)	0.0025	0.005	0.0025	0.005	0.0025	0.0025
Part Type 2	Throughput	0.2878	0.1638	0.0298	0	0.2371	0.1161
	Utilization	0.5533	0.315	0.0382	0	0.0228	0.0279
	Peak expected number of parts	15.7156	11.0661	10.2168	0	1.101	2.132
	M/C Rate (units/time)	0.02	0.02	0.03	0.2	0.4	0.16
Part Type 3	Throughput	0.1199	0.3107	0.3576	0.5252	0.1664	0.0965
	Utilization	0.0102	0.0265	0.0306	0.404	0.3197	0.1483
	Peak expected number of parts	0.2716	0.439	0.8172	7.1734	8.9672	0.754
	M/C Rate (units/time)	0.9	0.9	0.9	0.1	0.04	0.05

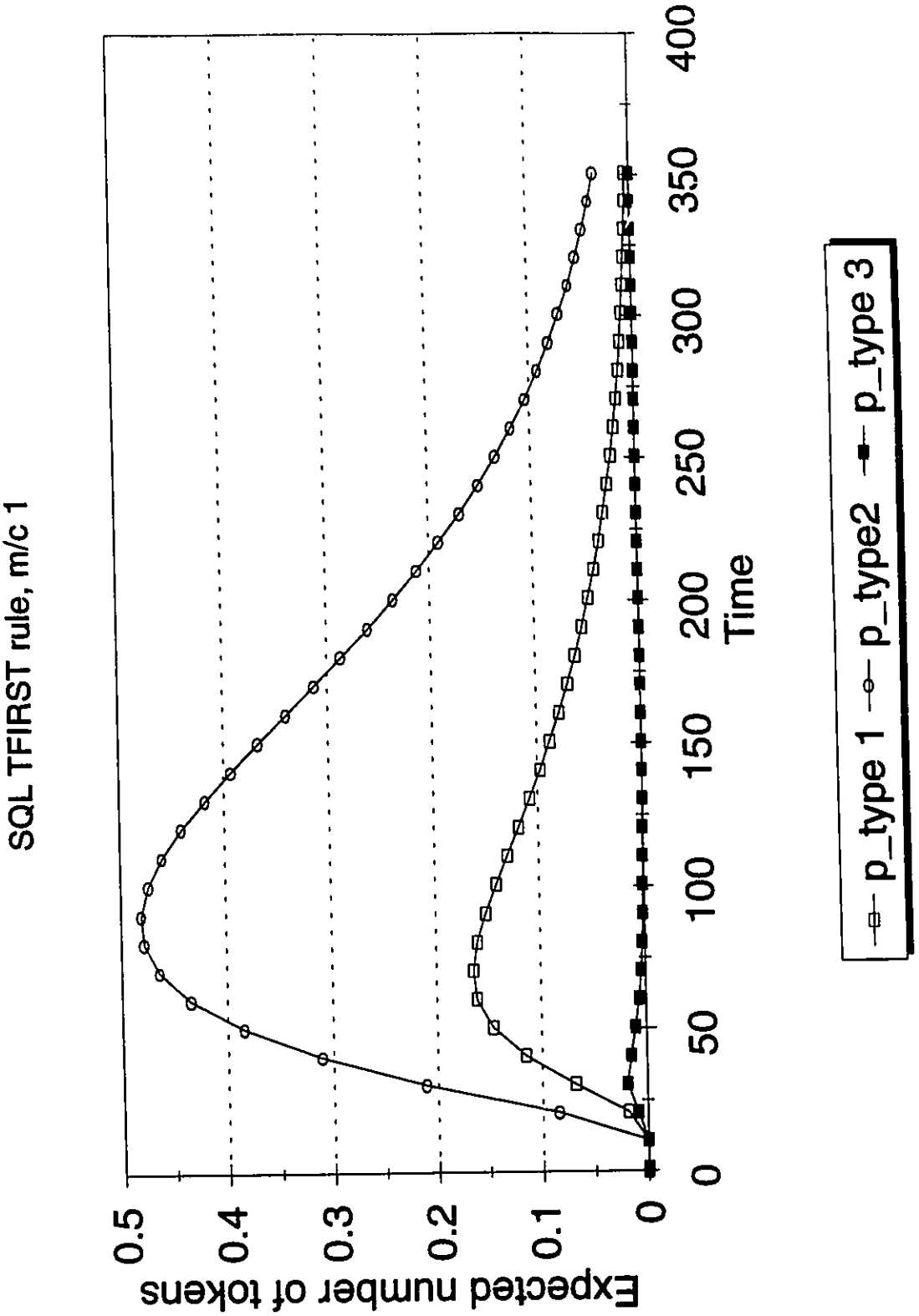


Fig 6.1 Loading pattern of machine 1 in Model I

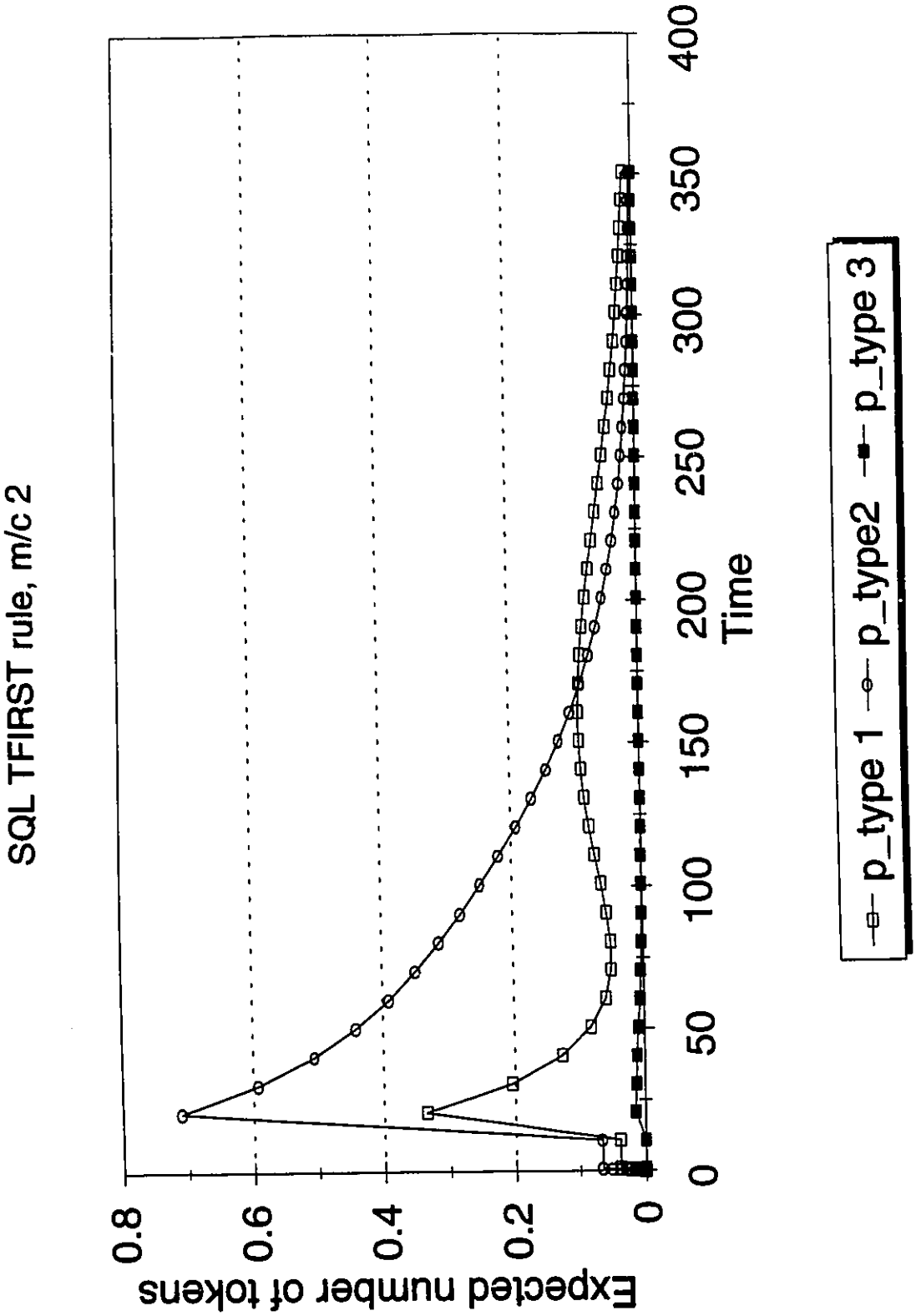


Fig 6.2 Loading pattern of machine 2 in Model I

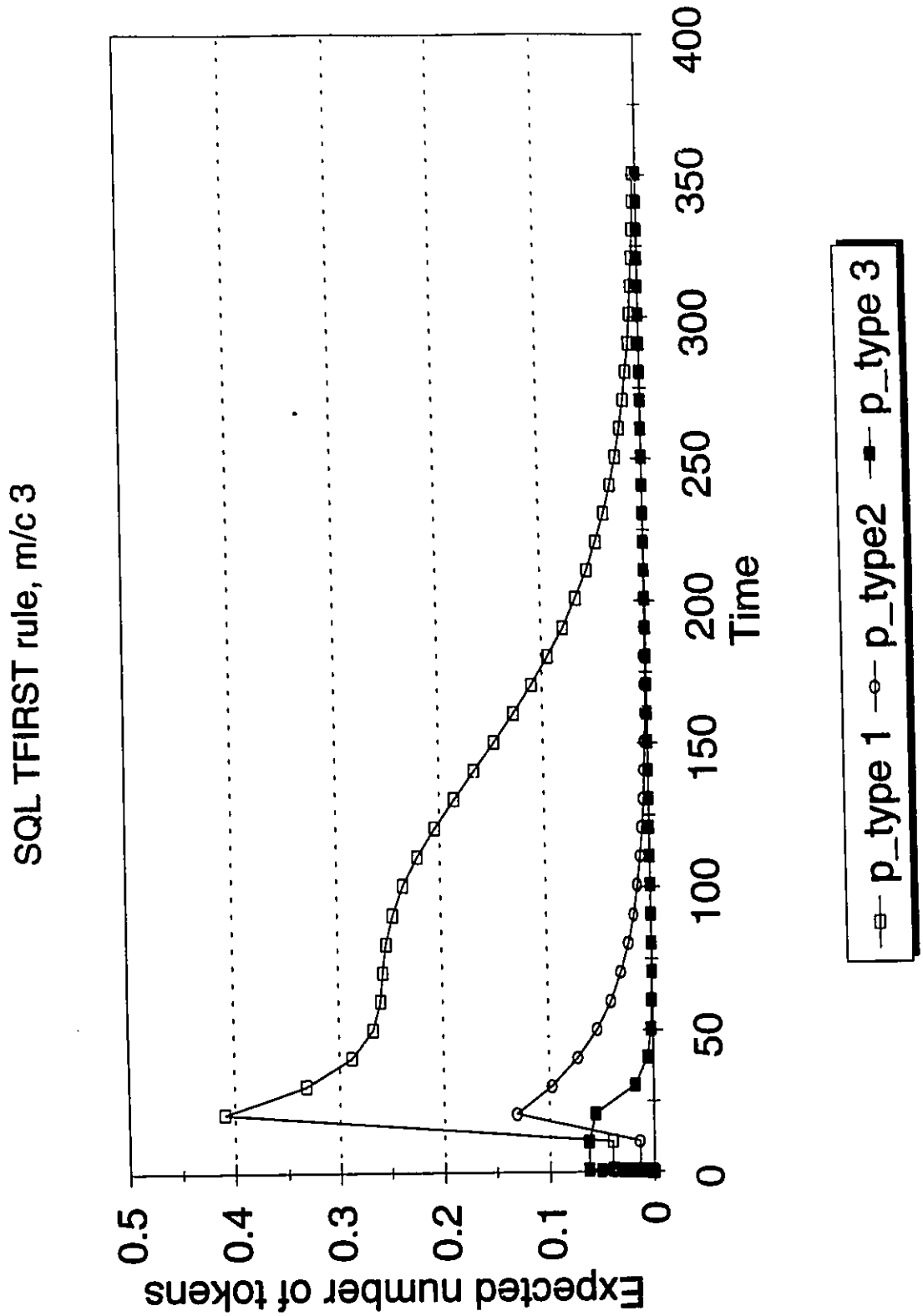


Fig 6.3 Loading pattern of machine 3 in Model I

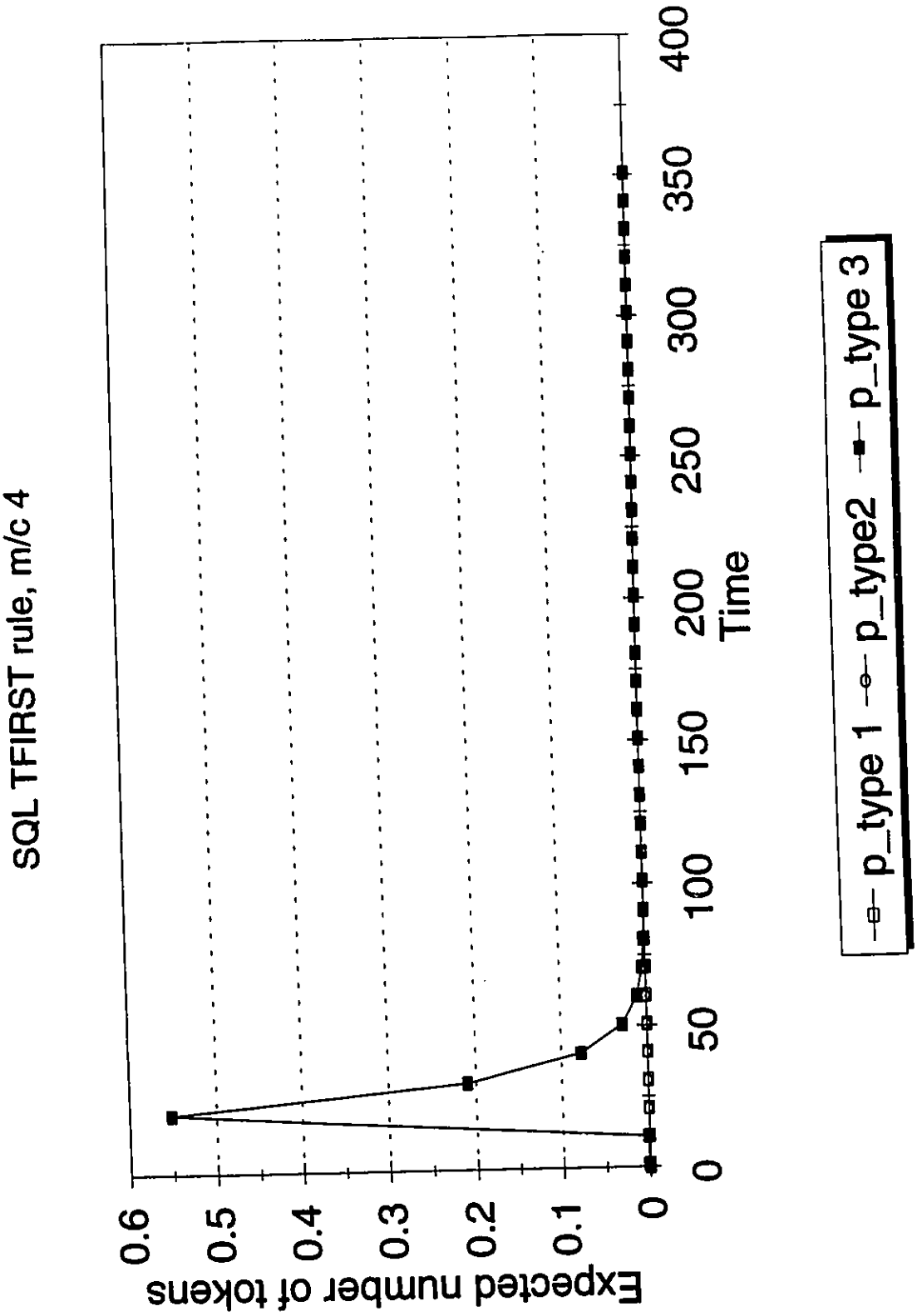


Fig 6.4 Loading pattern of machine 4 in Model I

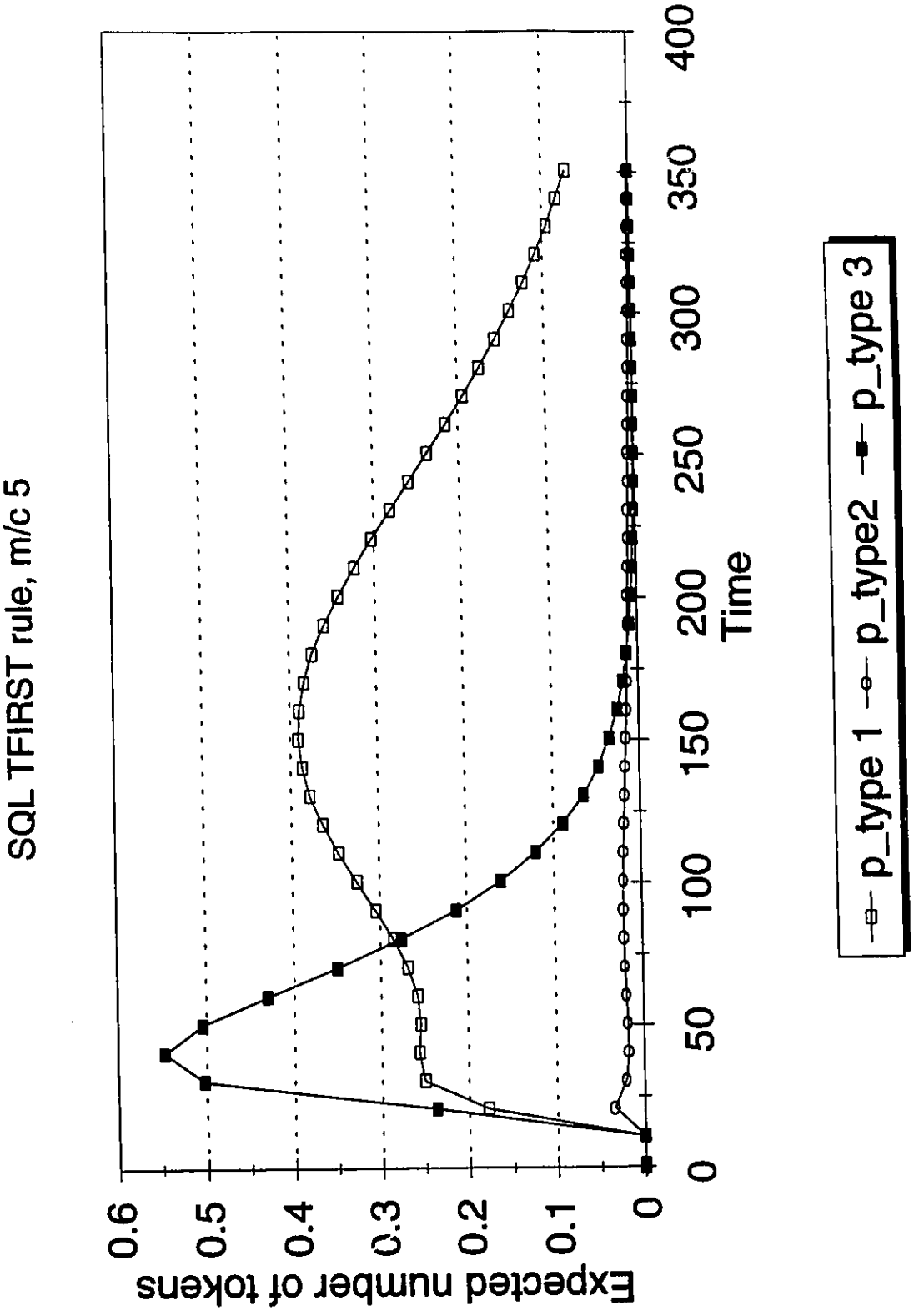


Fig 6.5 Loading pattern of machine 5 in Model I

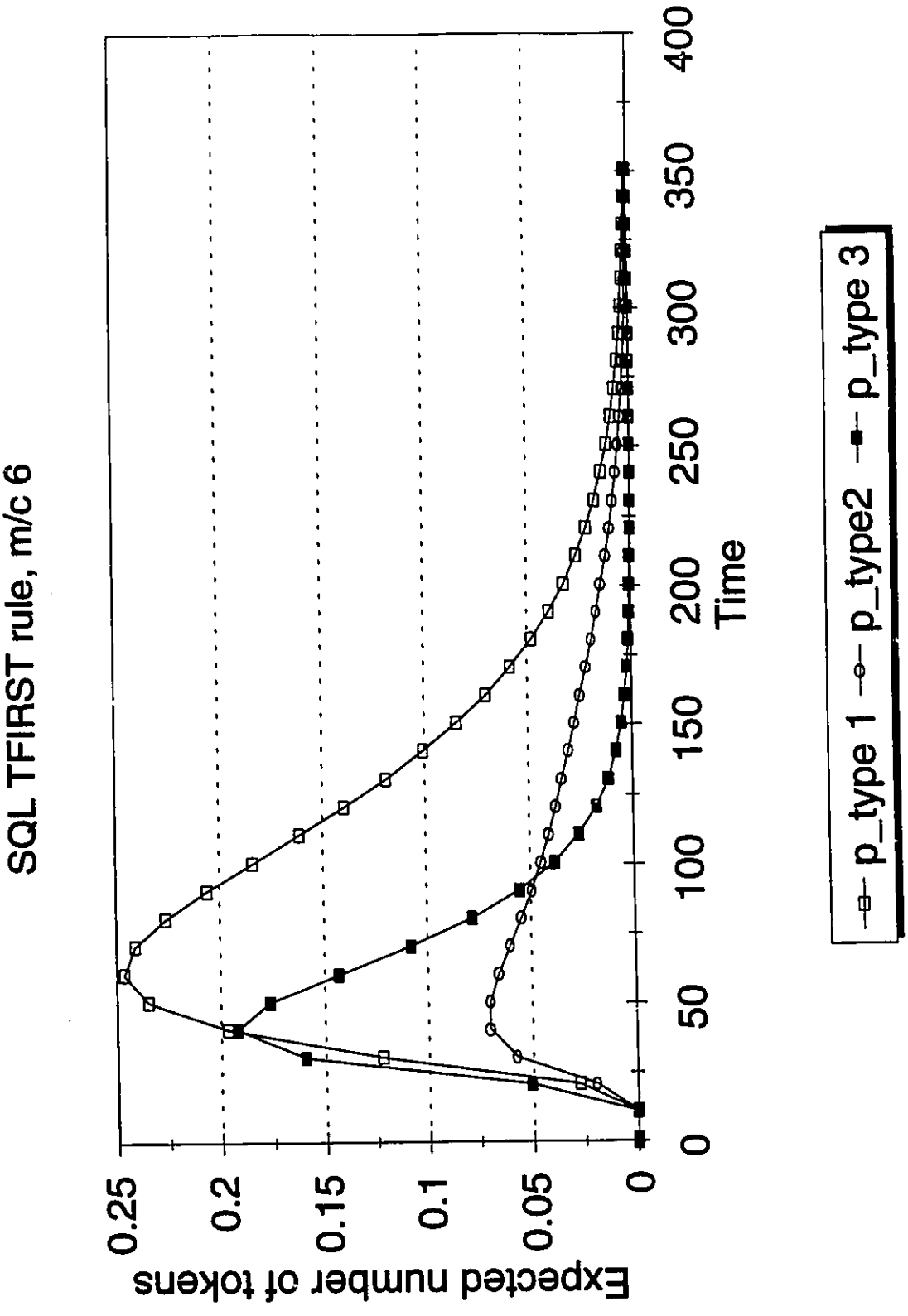


Fig 6.6 Loading pattern of machine 6 in Model I

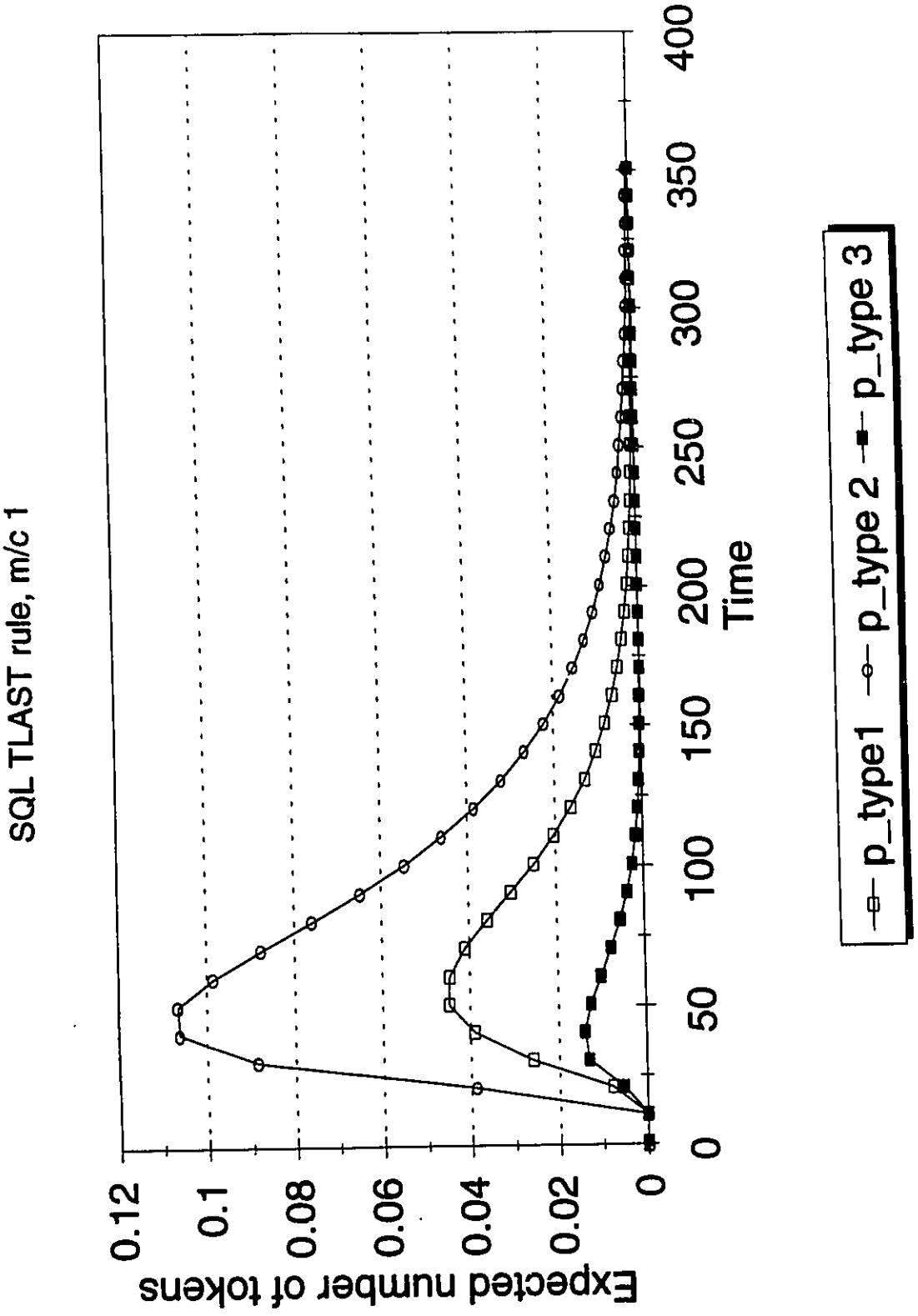


Fig 6.7 Loading pattern of machine 1 in Model II

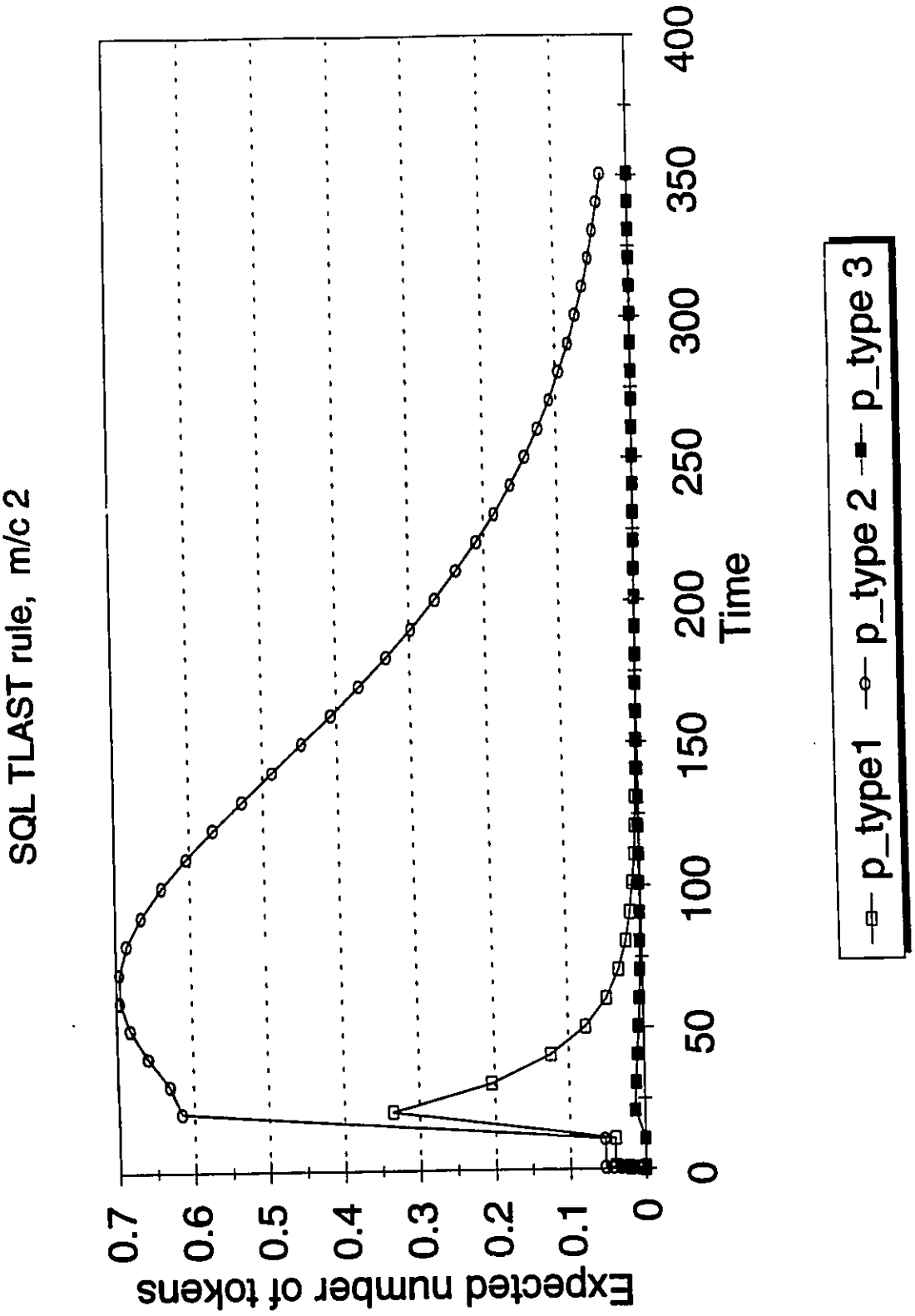


Fig 6.8 Loading pattern of machine 2 in Model II

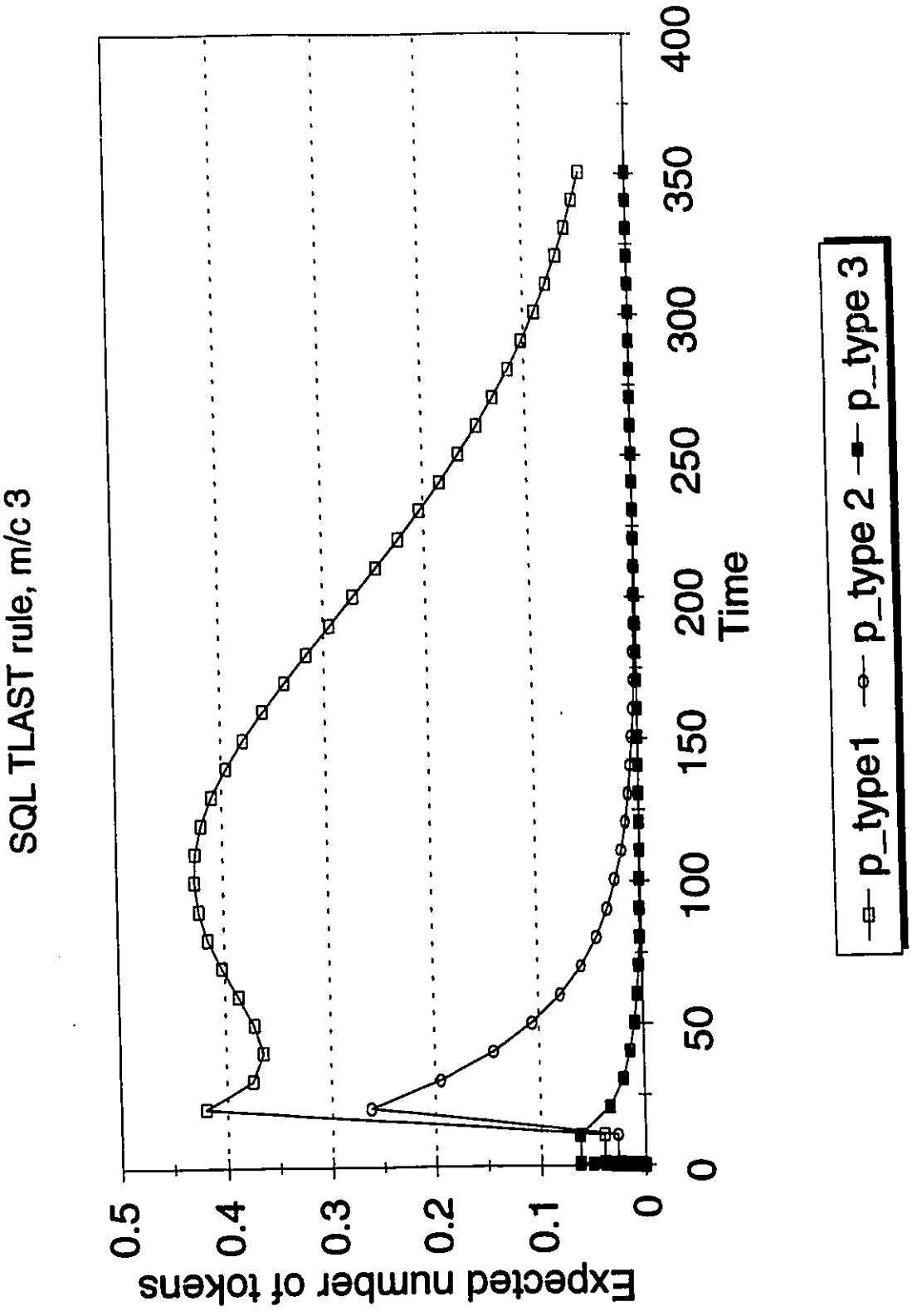


Fig 6.9 Loading pattern of machine 3 in Model II

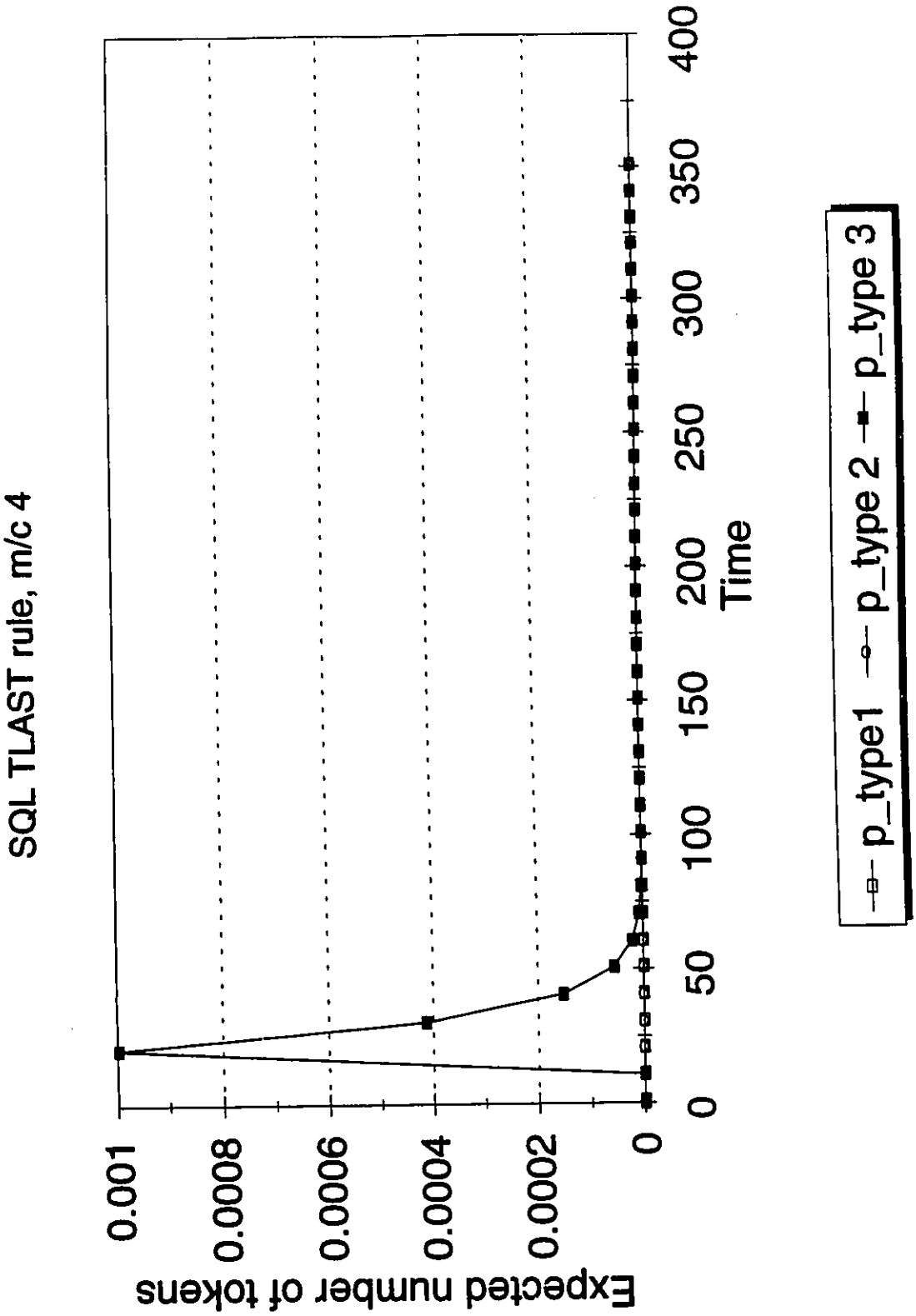


Fig 6.10 Loading pattern of machine 4 in Model II

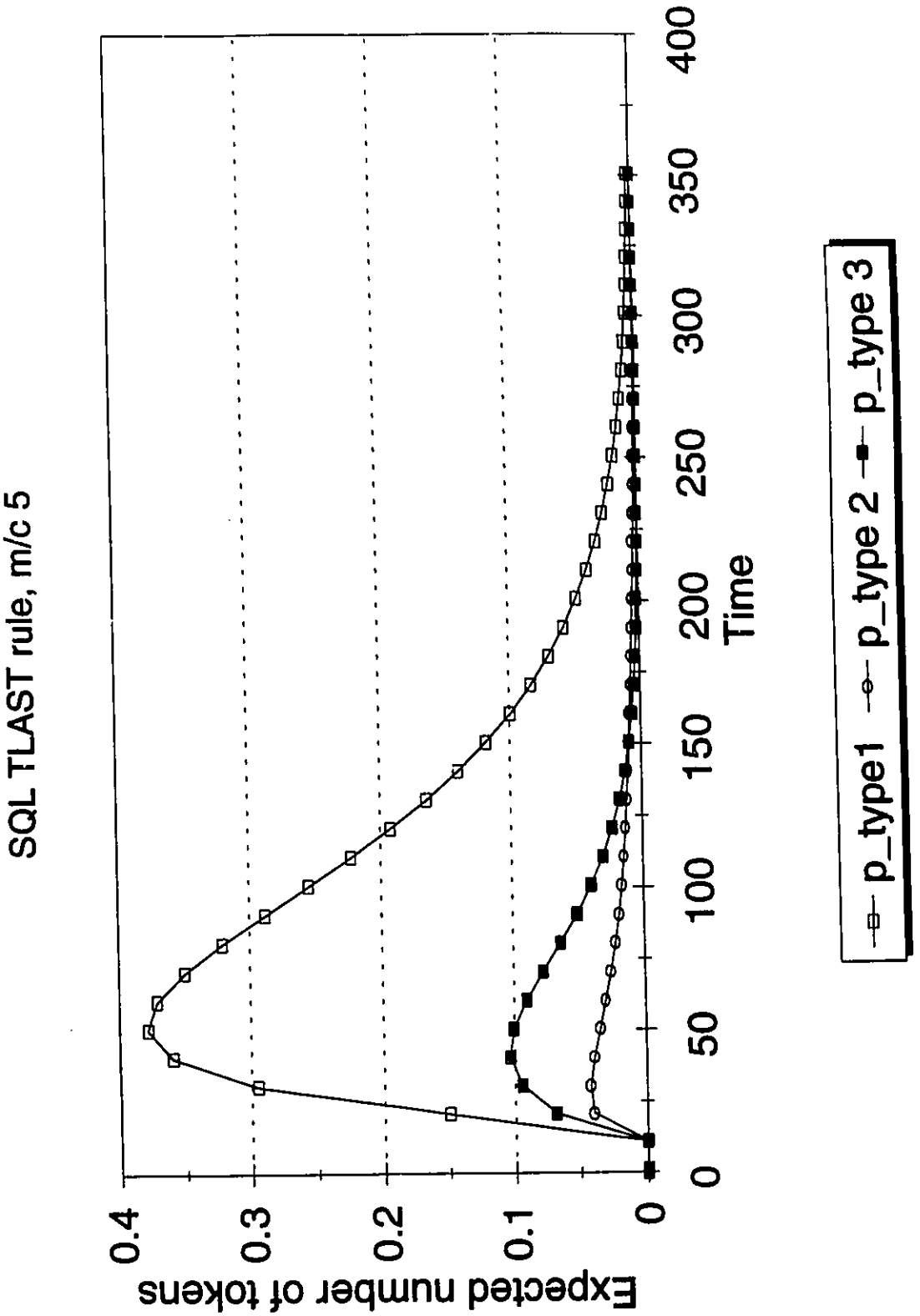


Fig 6.11 Loading pattern of machine 5 in Model II

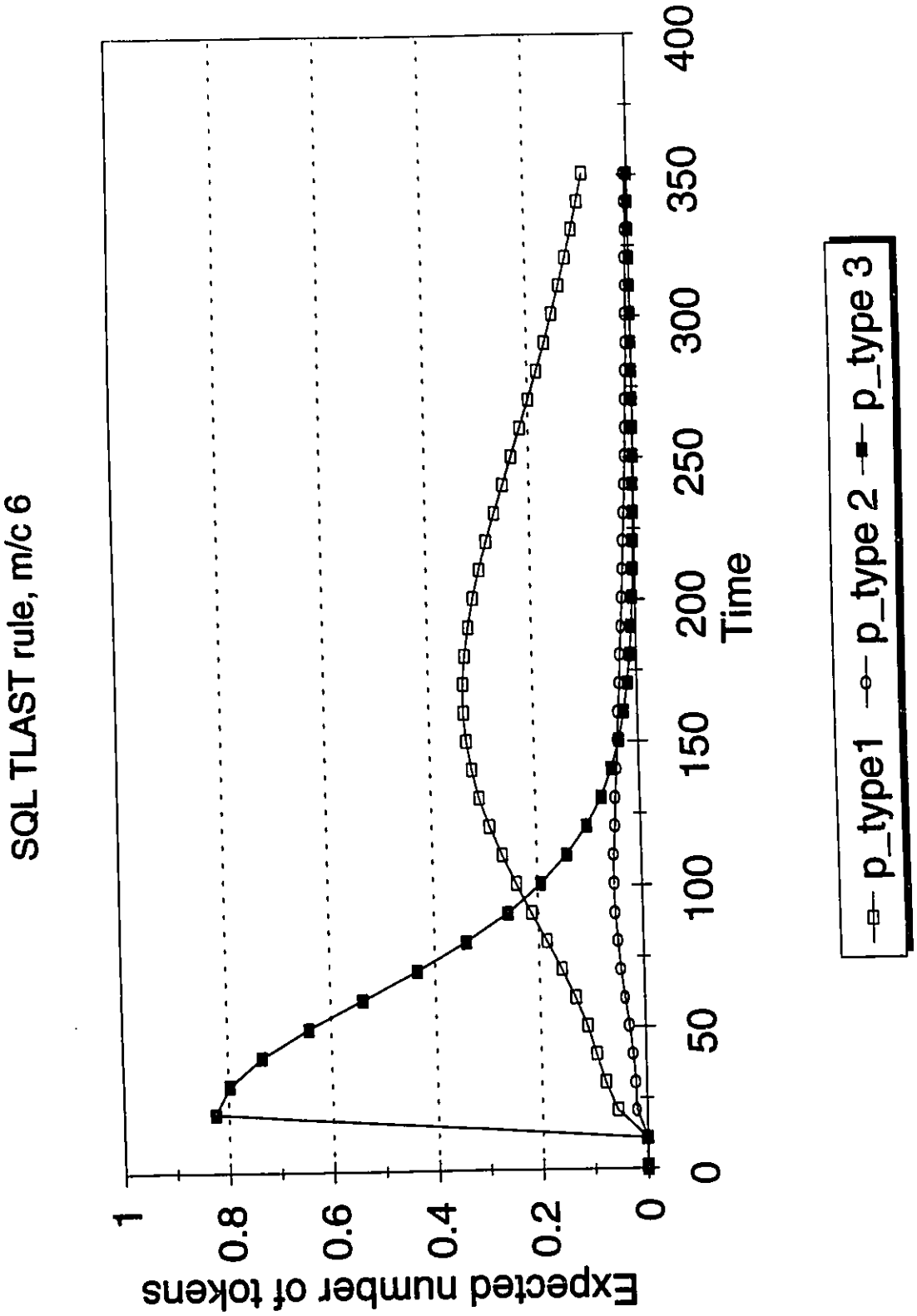


Fig 6.12 Loading pattern of machine 6 in Model II

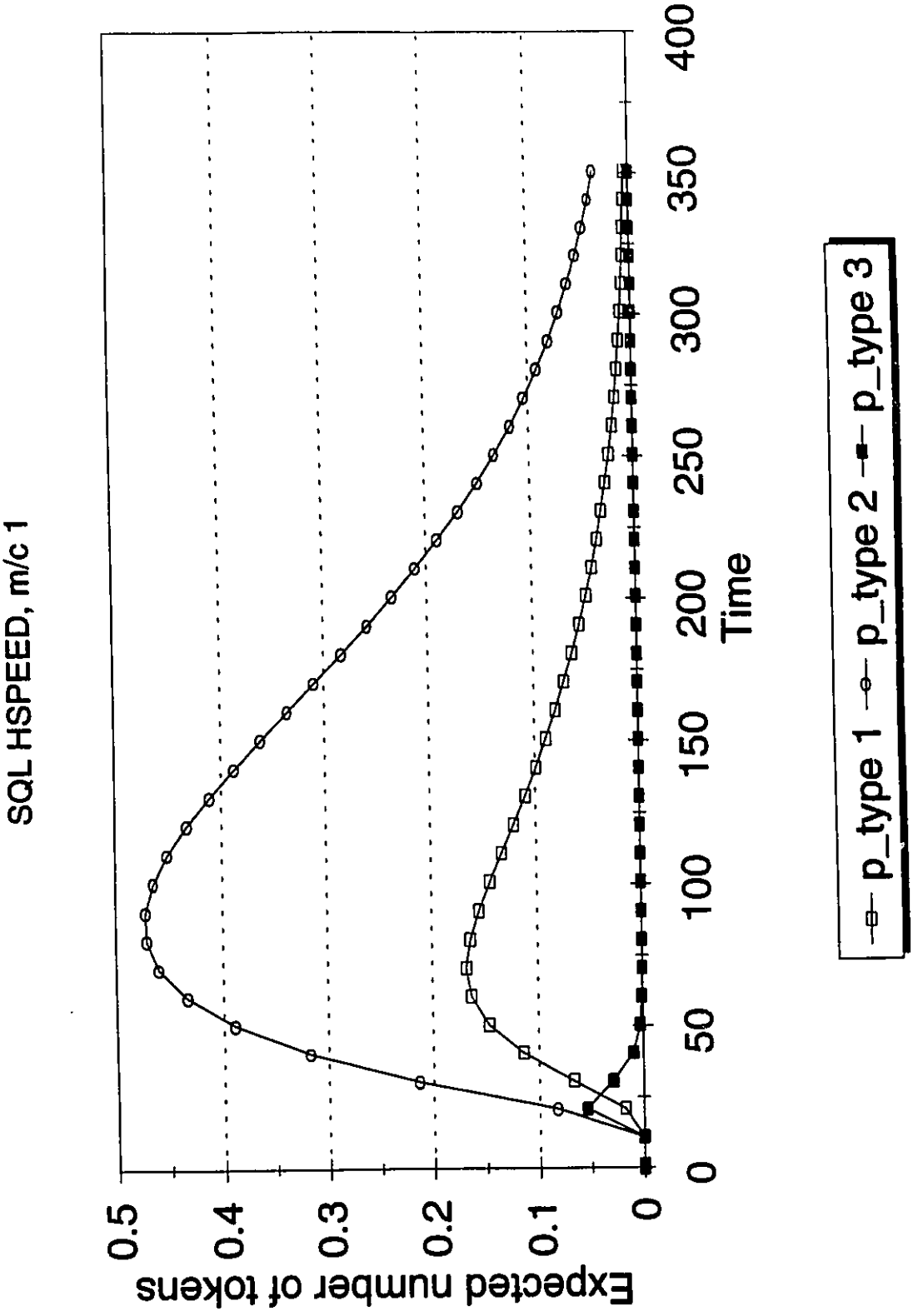


Fig 6.13 Loading pattern of machine 1 in Model III

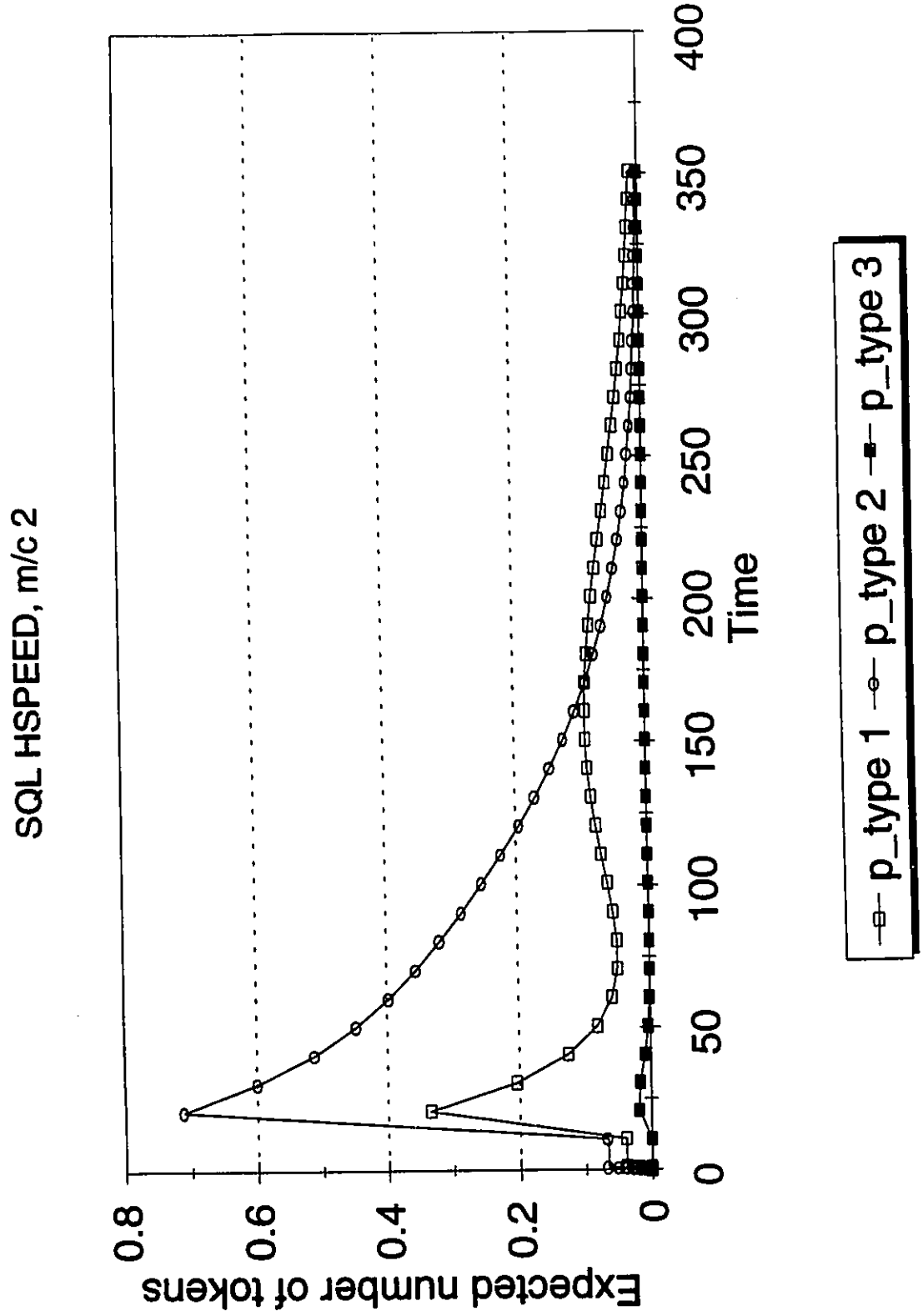


Fig 6.14 Loading pattern of machine 2 in Model III

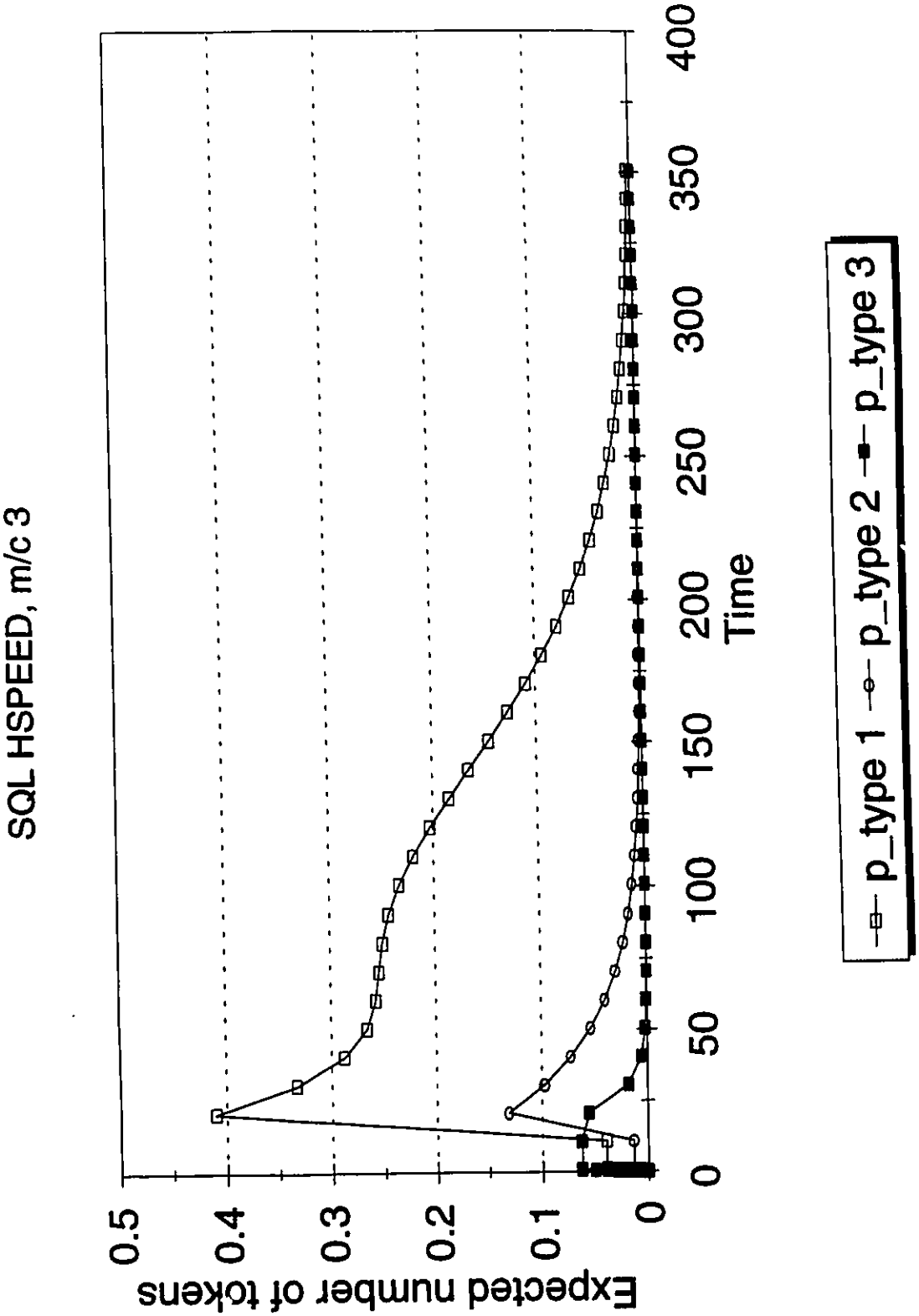


Fig 6.15 Loading pattern of machine 3 in Model III

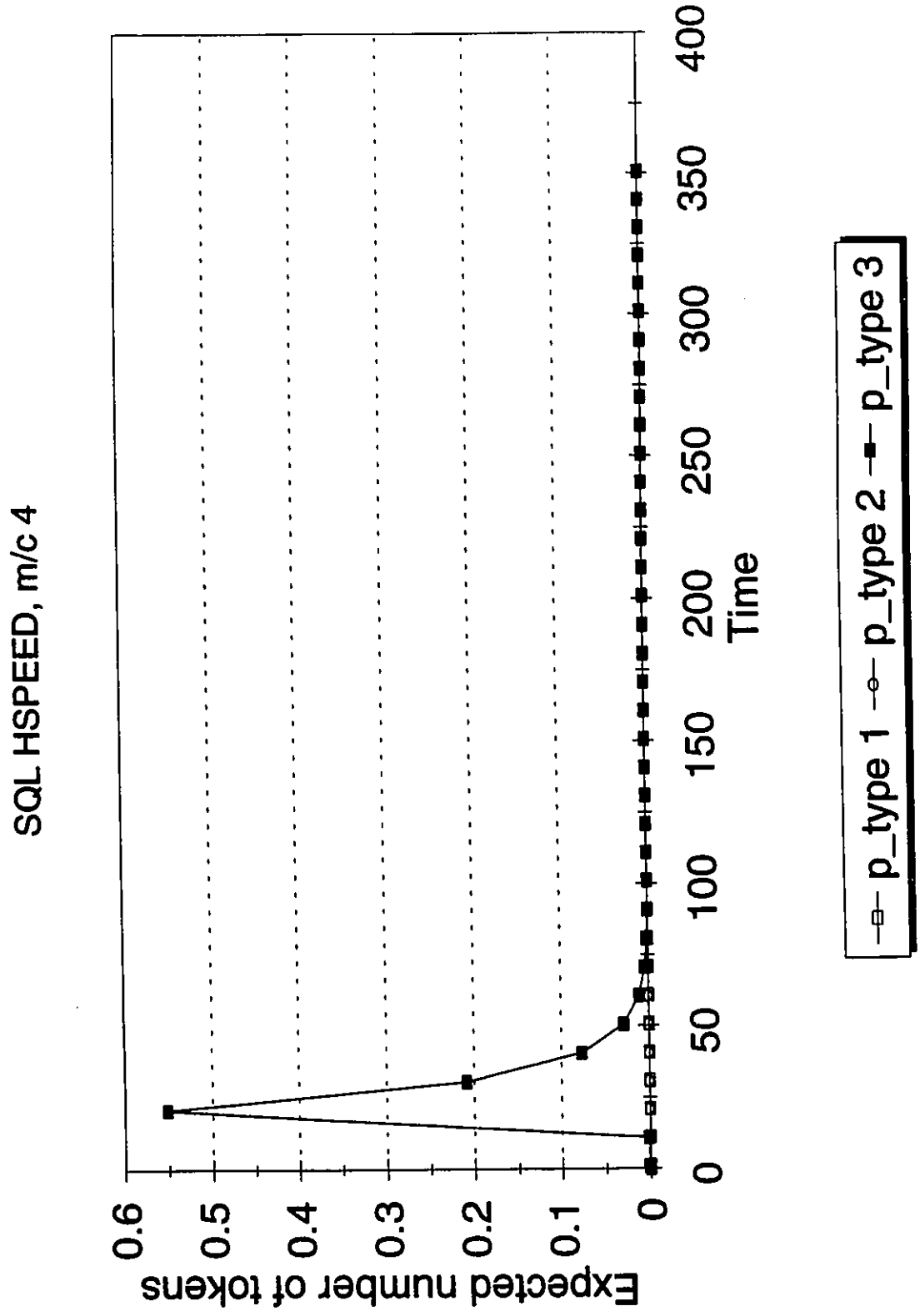


Fig 6.16 Loading pattern of machine 4 in Model III

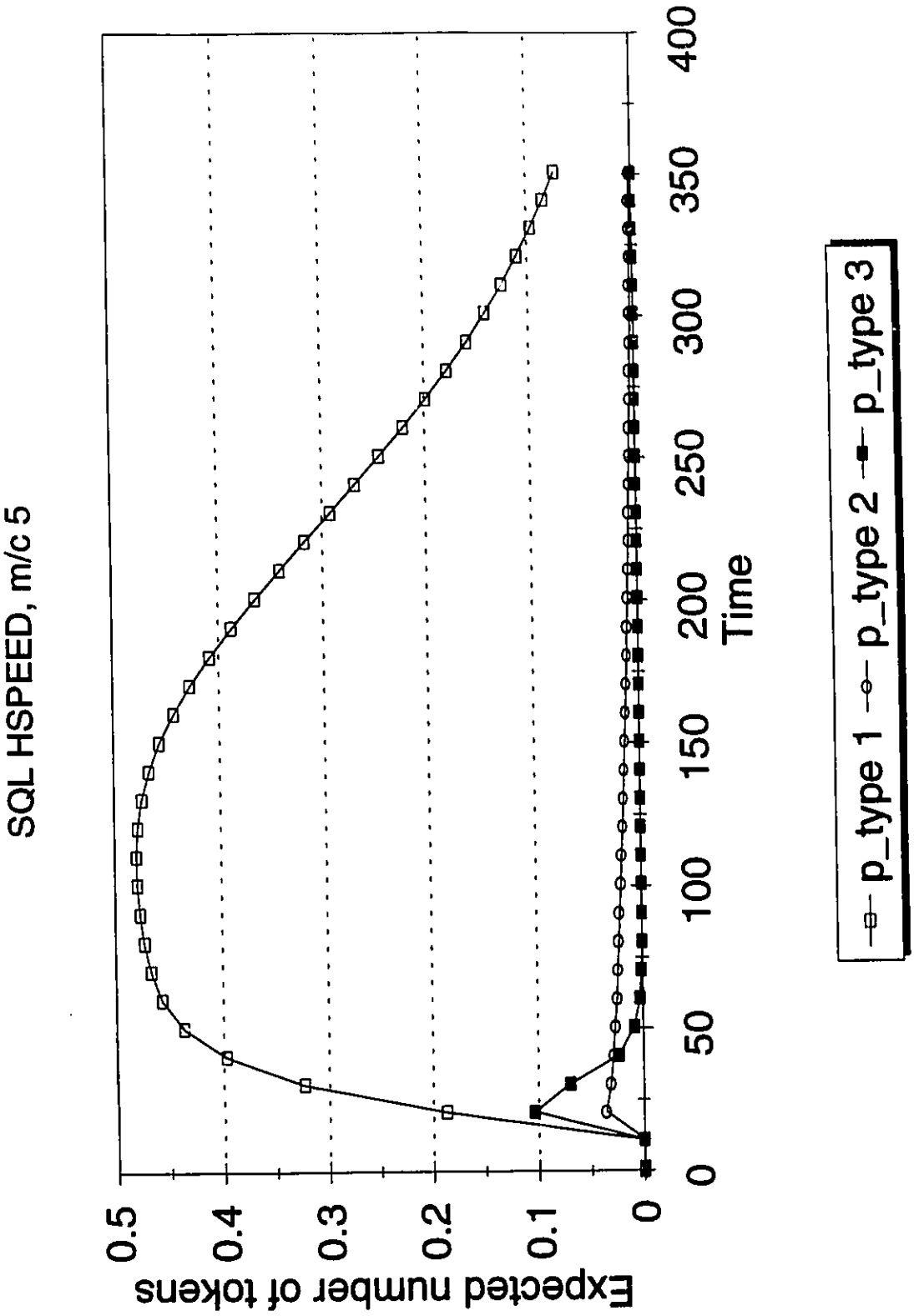


Fig 6.17 Loading pattern of machine 5 in Model III

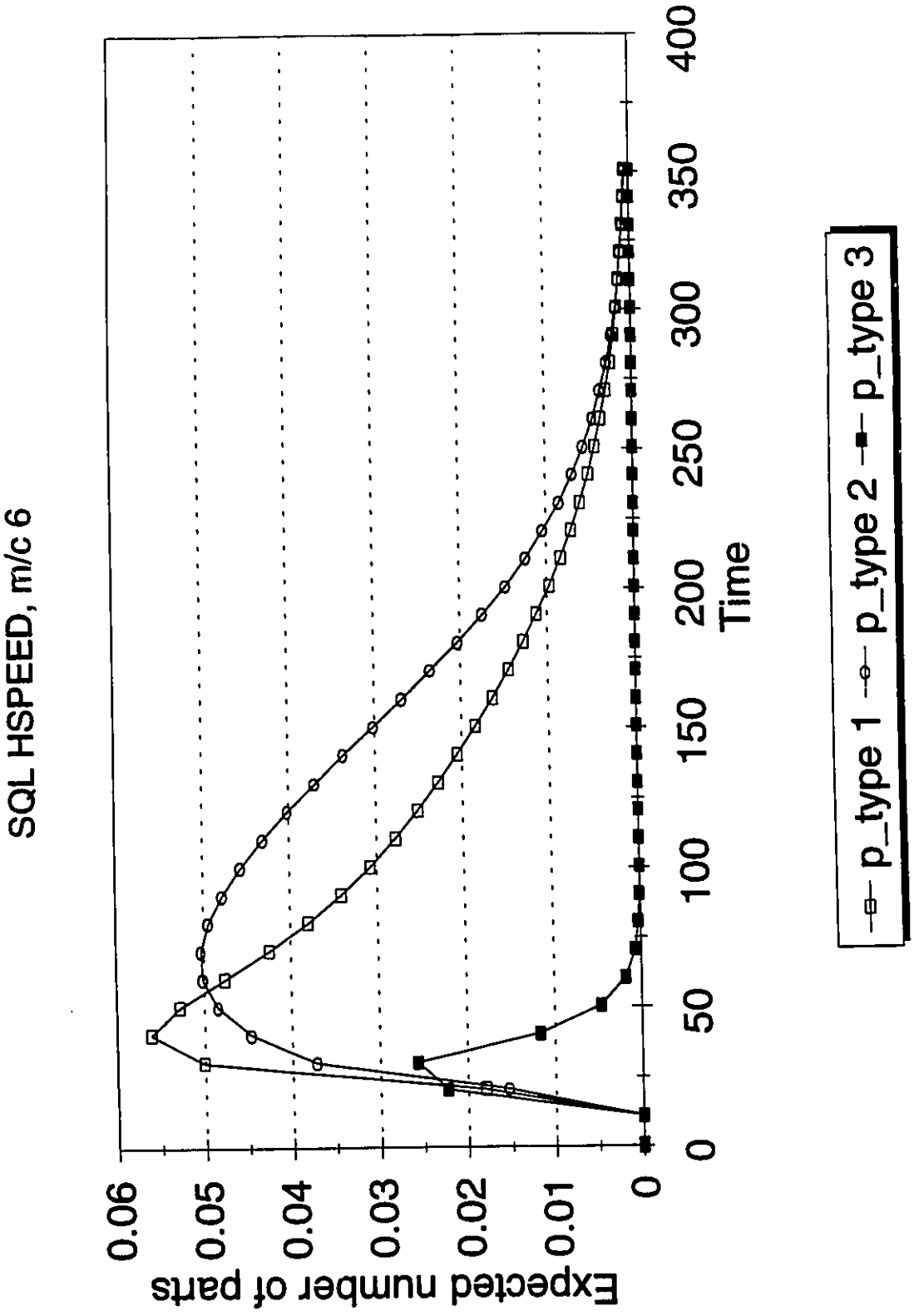


Fig 6.18 Loading pattern of machine 6 in Model III

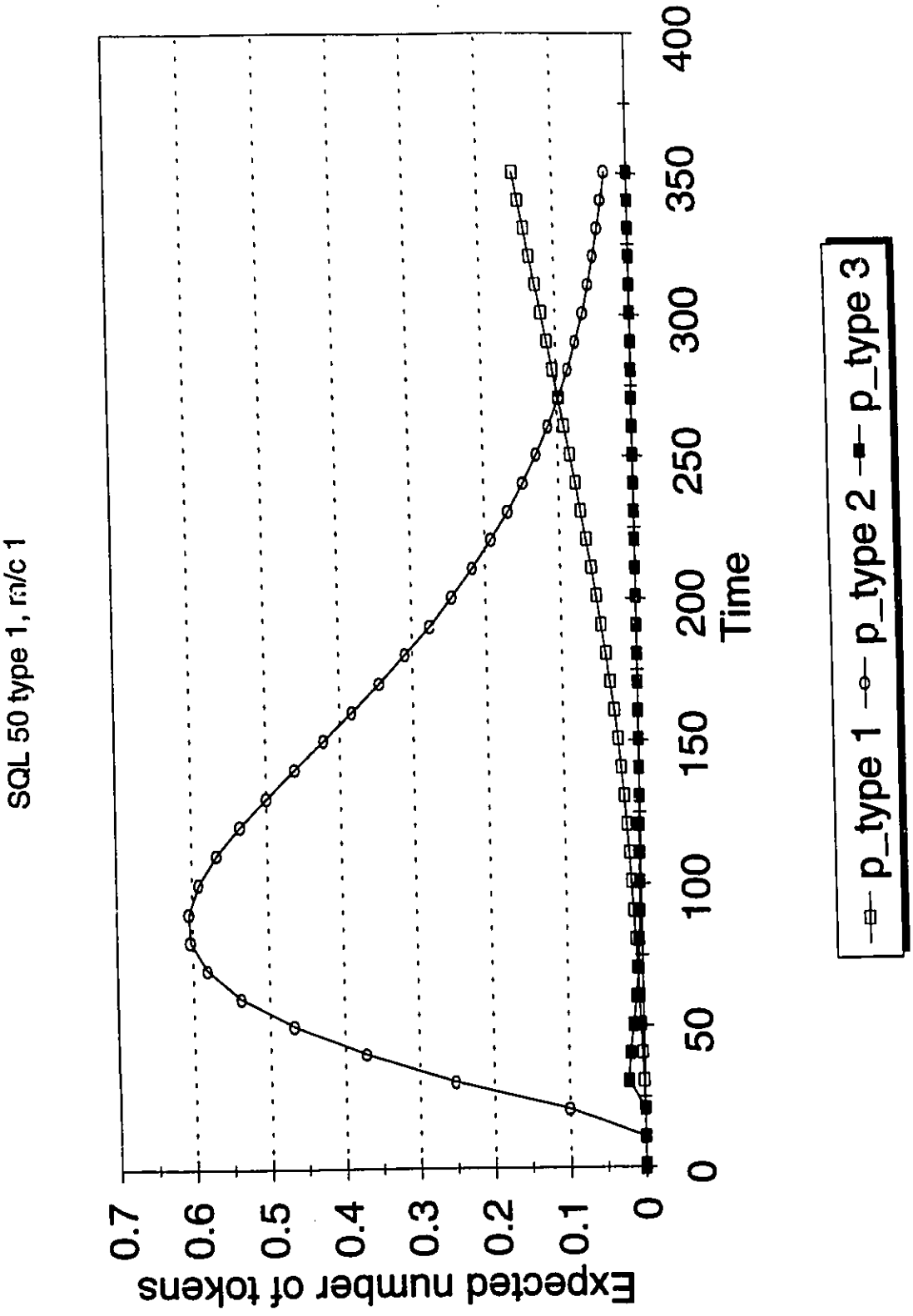


Fig 6.19 Loading pattern of machine 1 in Model IV

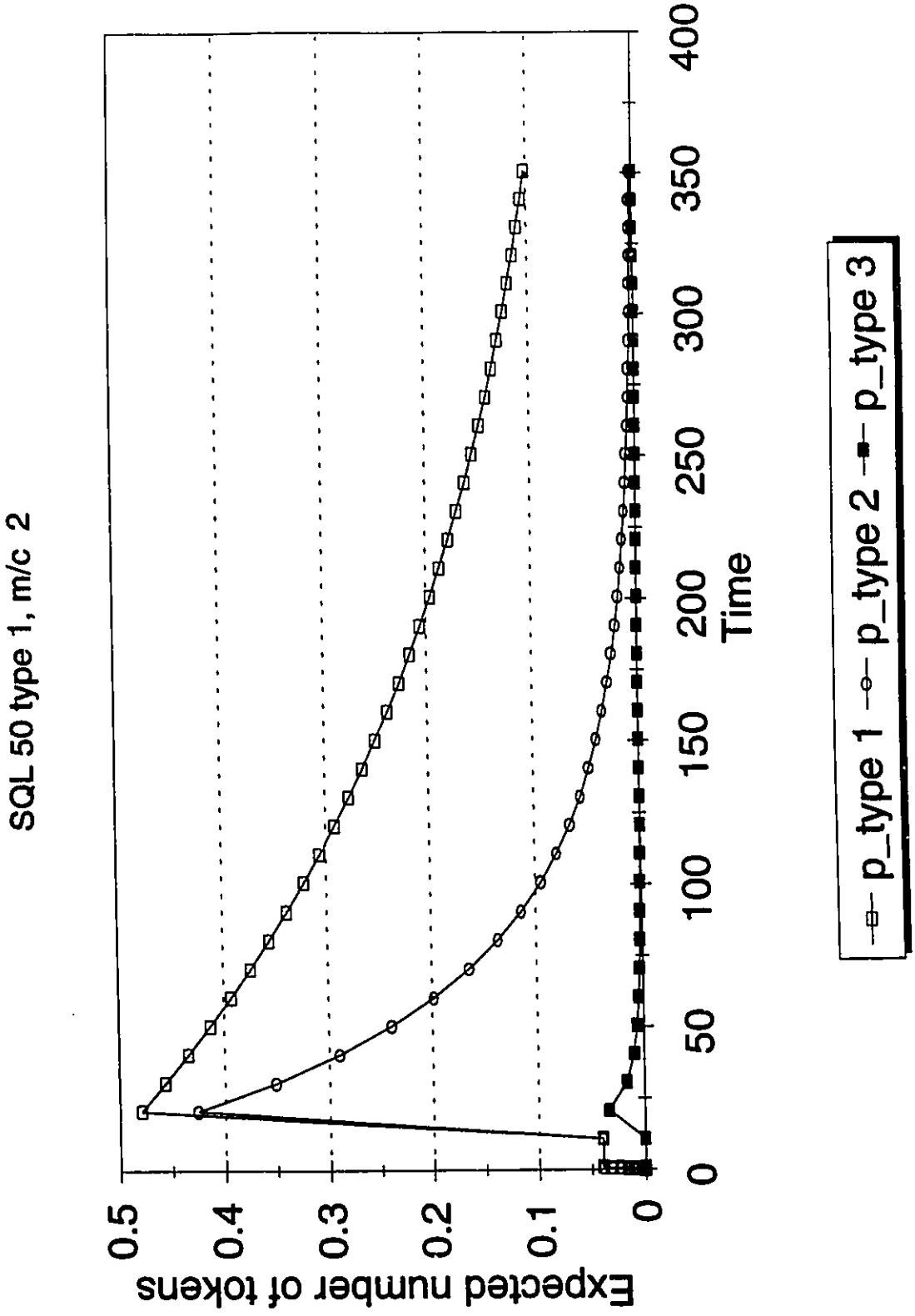


Fig 6.20 Loading pattern of machine 2 in Model IV

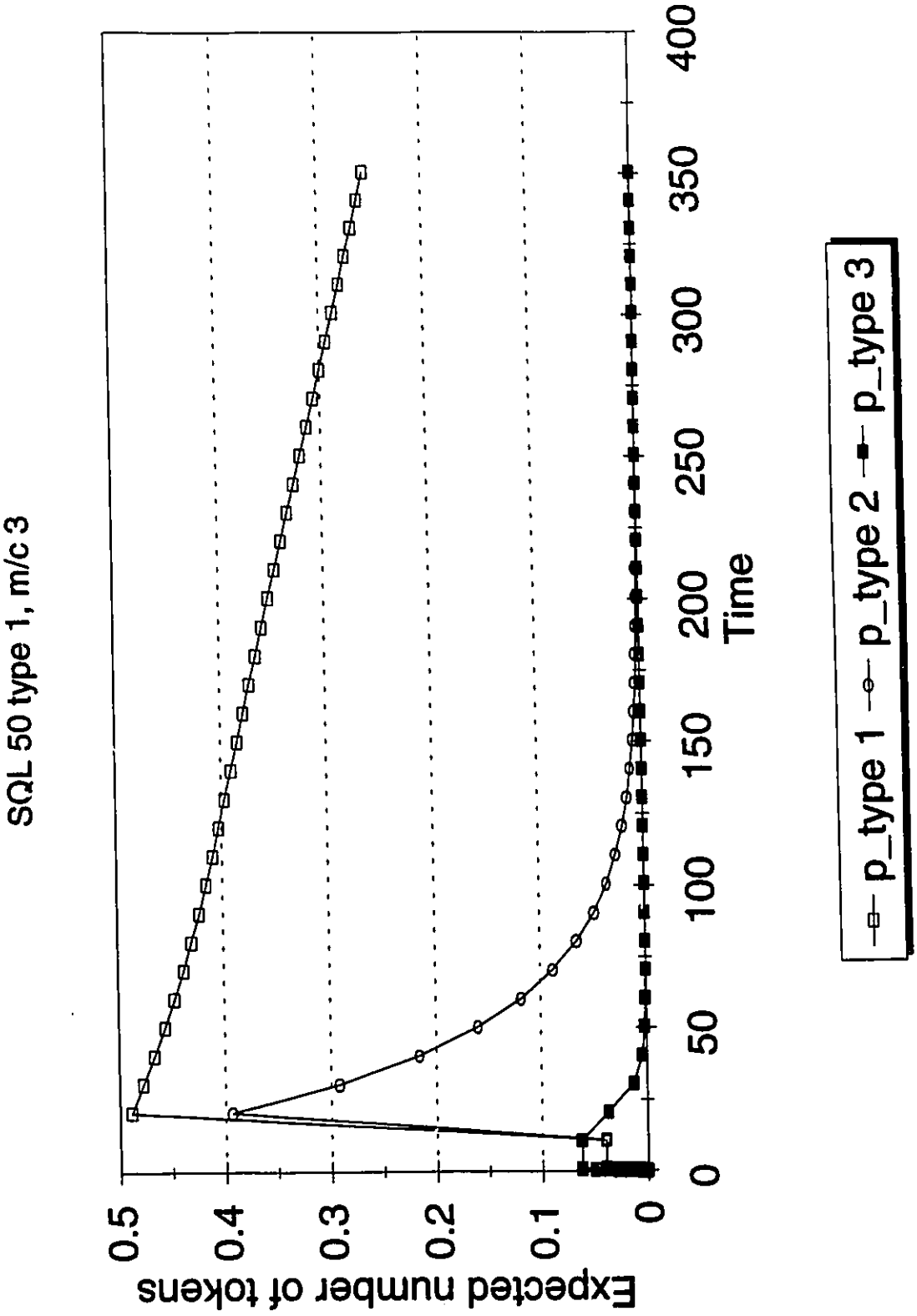


Fig 6.21 Loading pattern of machine 3 in Model IV

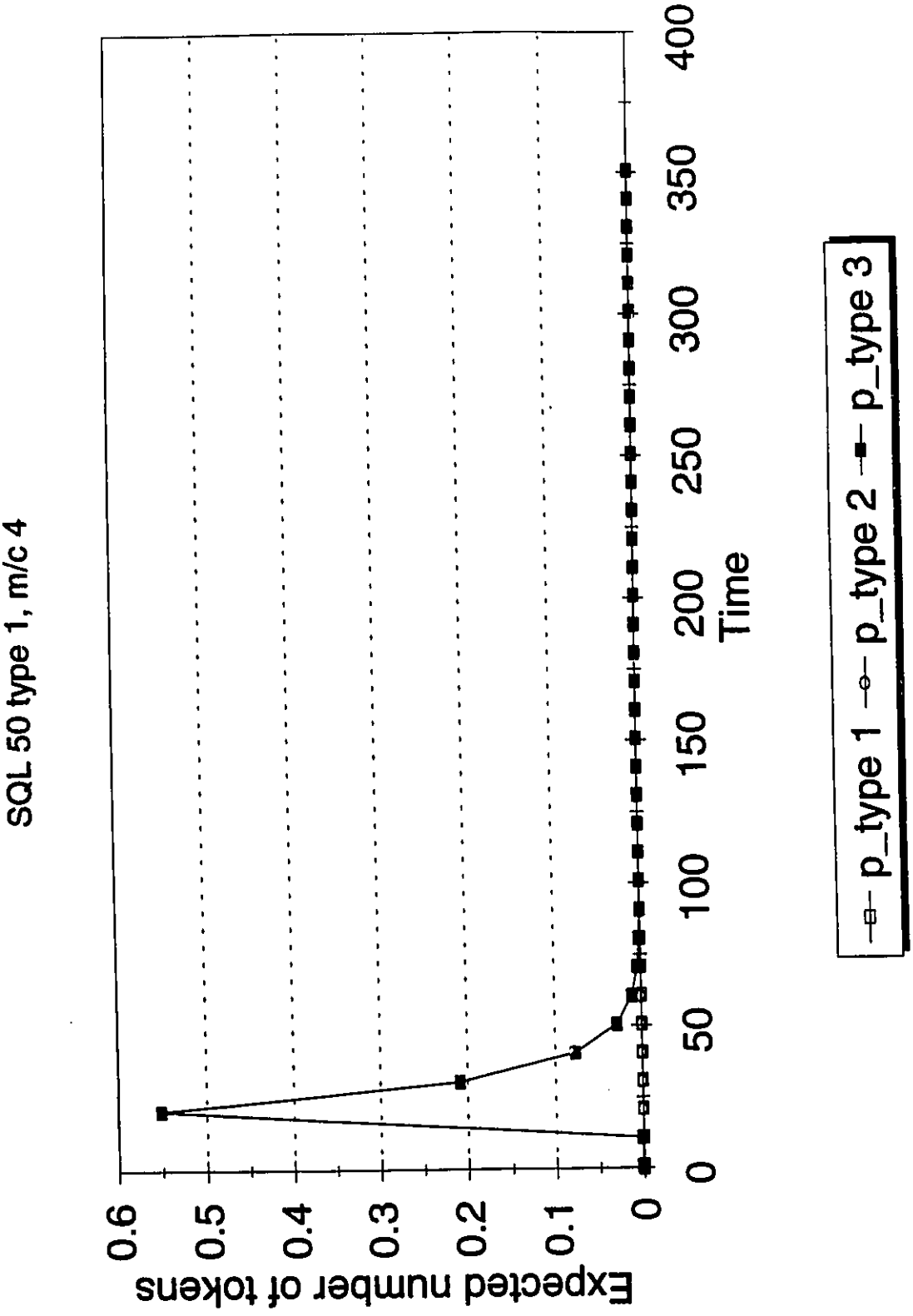


Fig 6.22 Loading pattern of machine 4 in Model IV

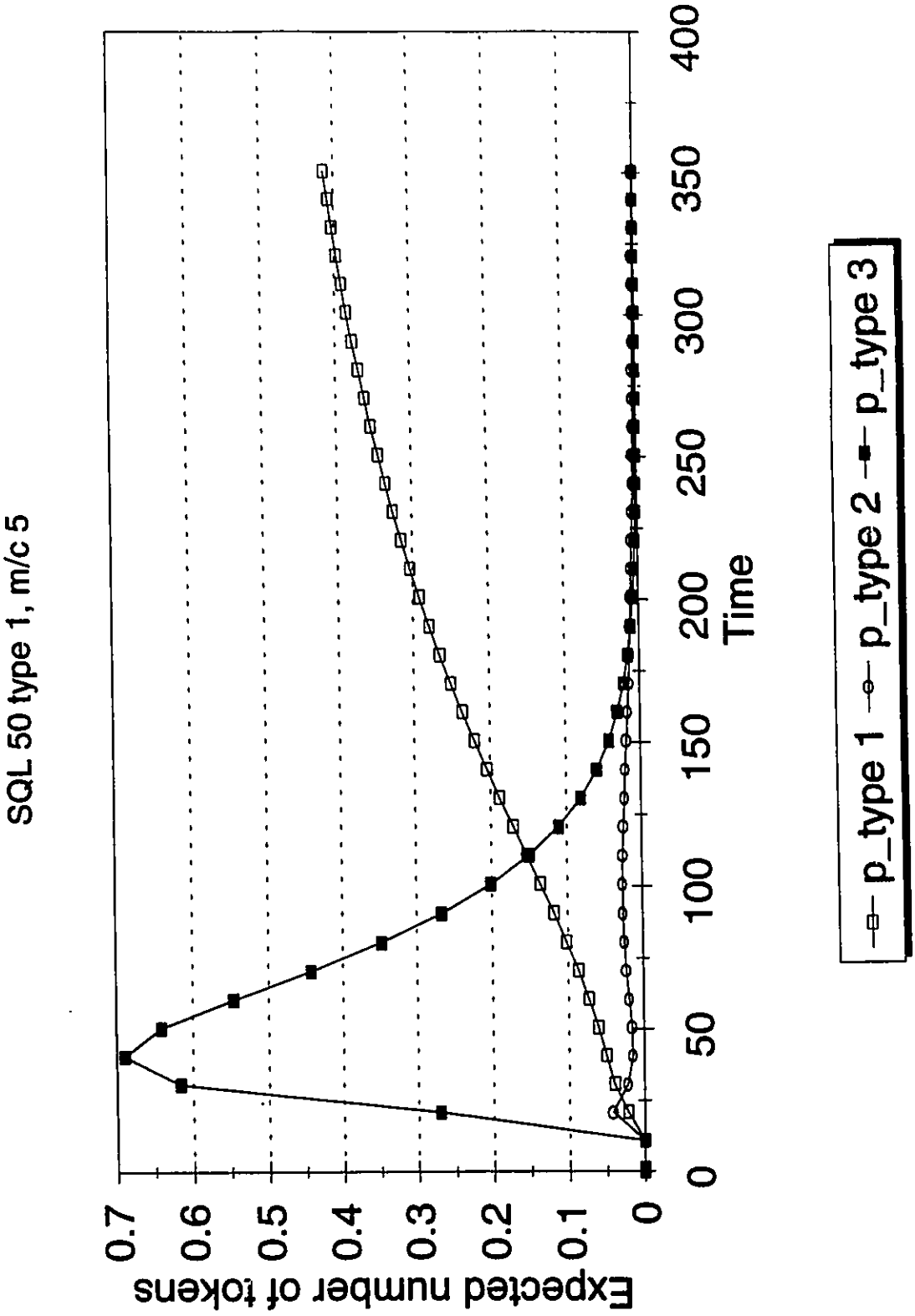


Fig 6.23 Loading pattern of machine 5 in Model IV

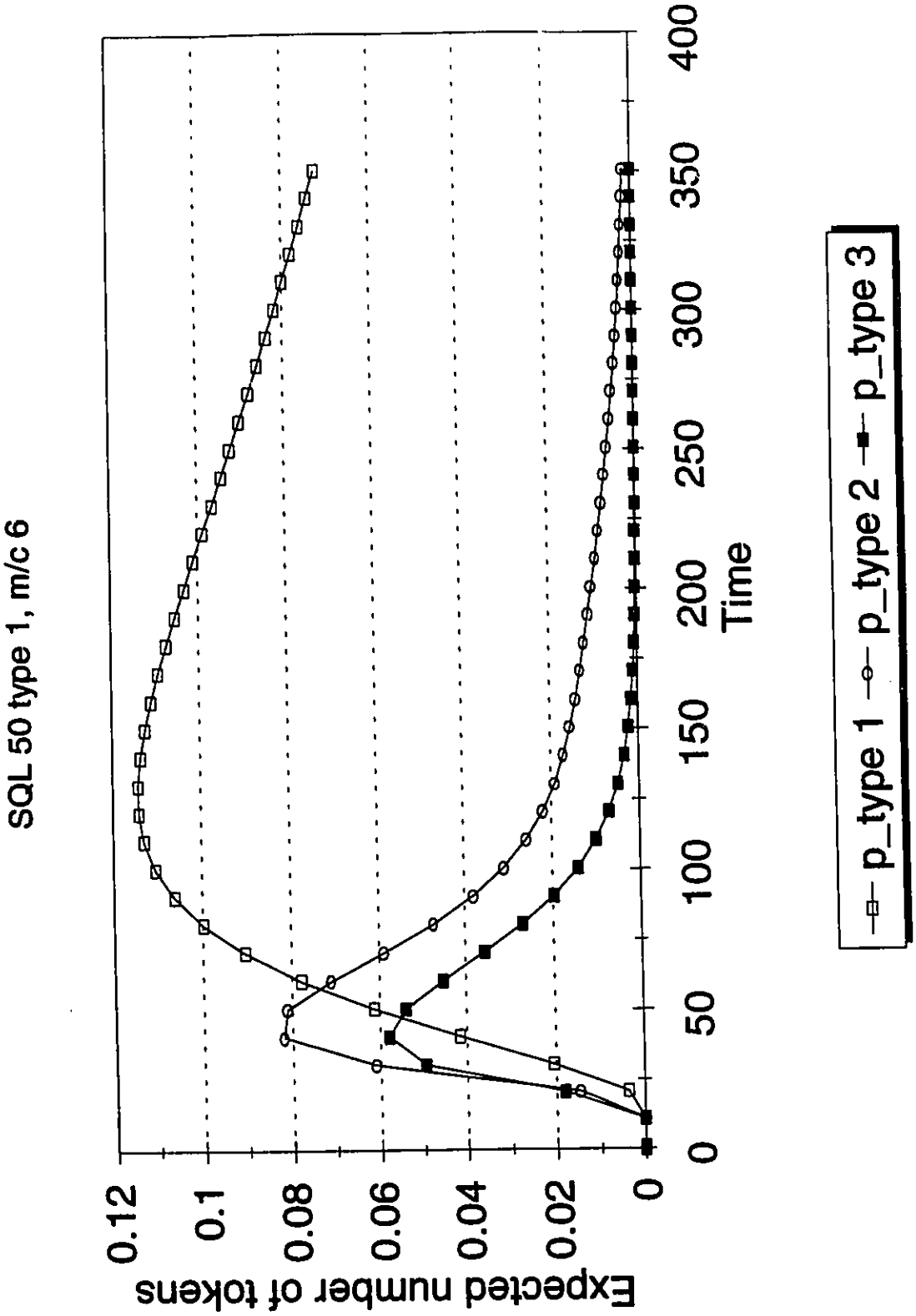


Fig 6.24 Loading pattern of machine 6 in Model IV

Model I and Model II comparison

Part Type 1

Throughput values

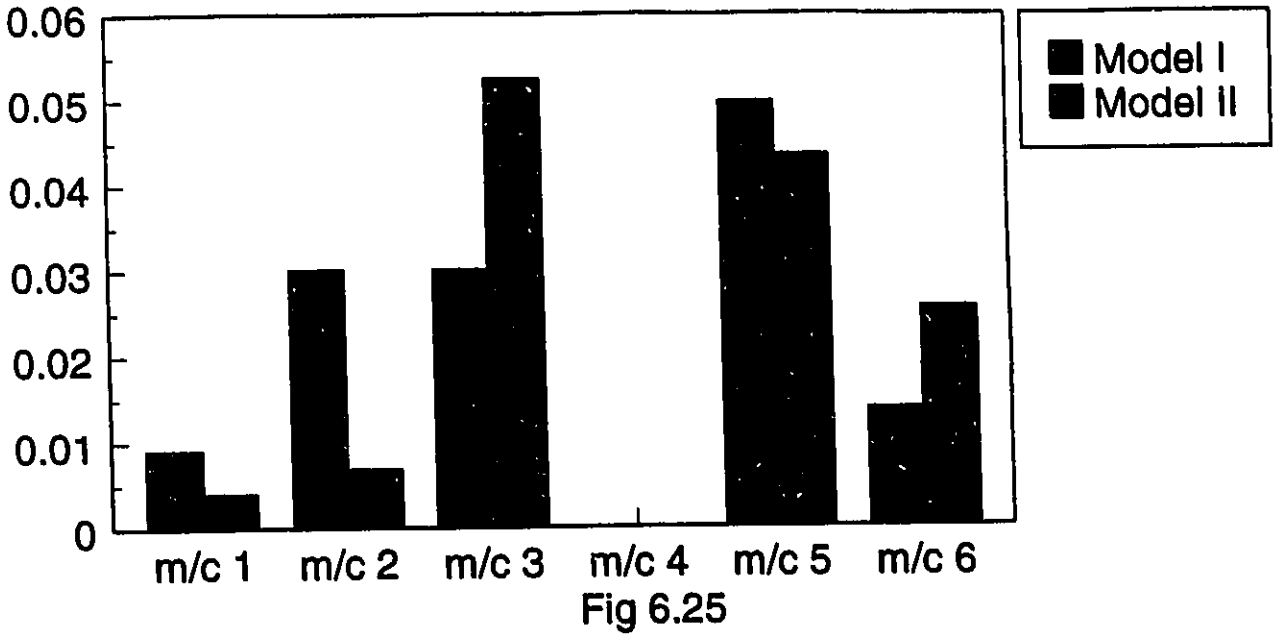


Fig 6.25

Part Type 1

Utilization values

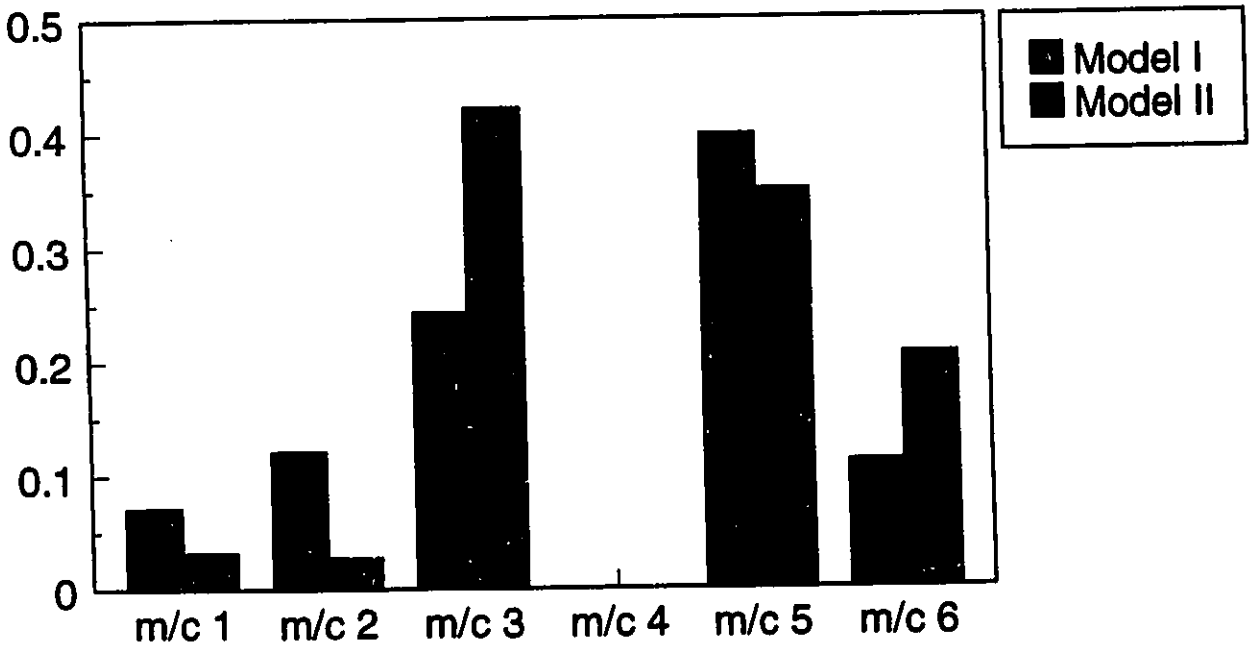


Fig 6.26

Model I and Model II comparison

Part Type 2

Throughput values

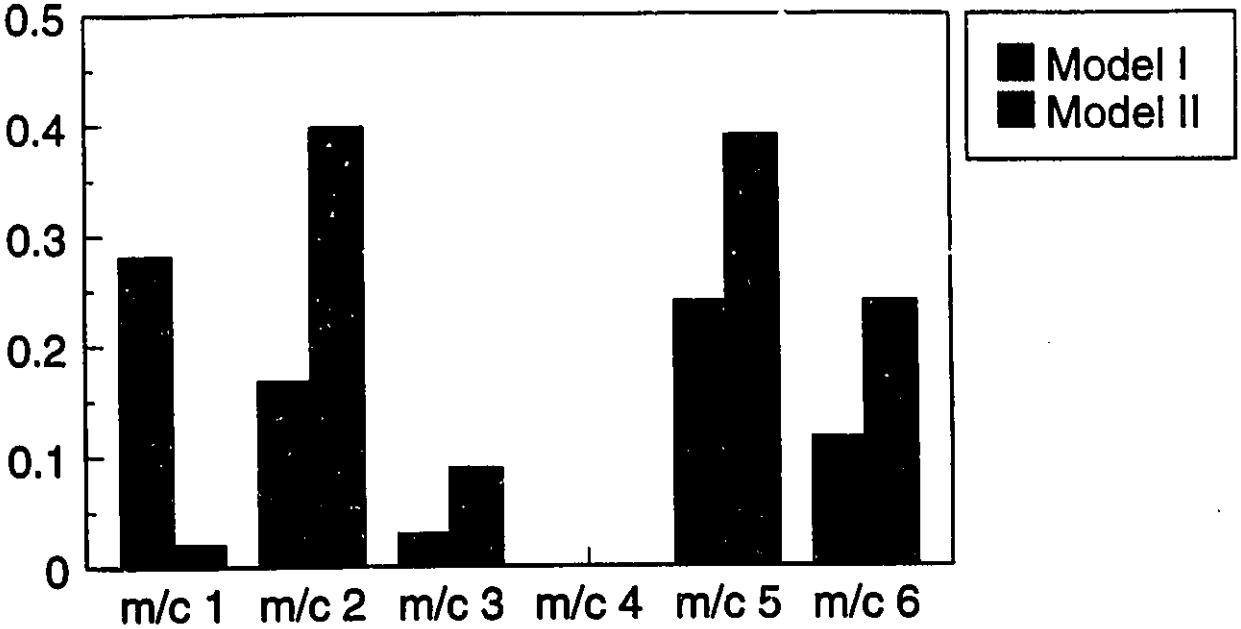


Fig 6.27

Part Type 2

Utilization values

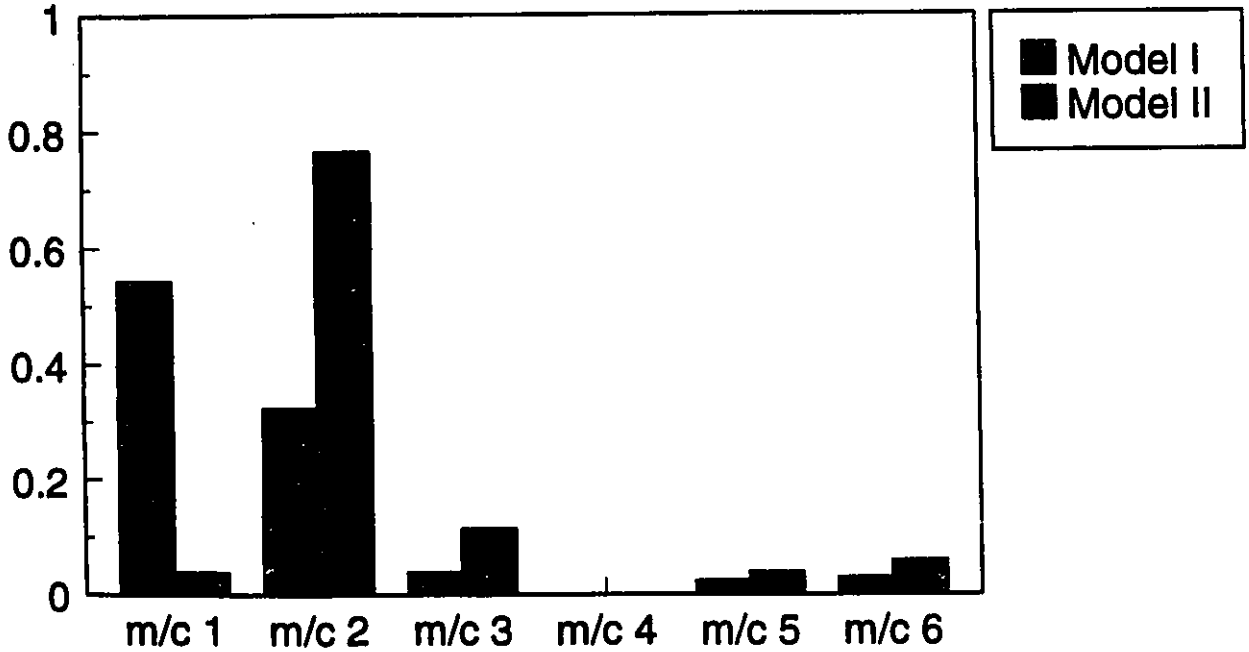


Fig 6.28

Model I and Model II comparison

Part Type 3

Throughput values

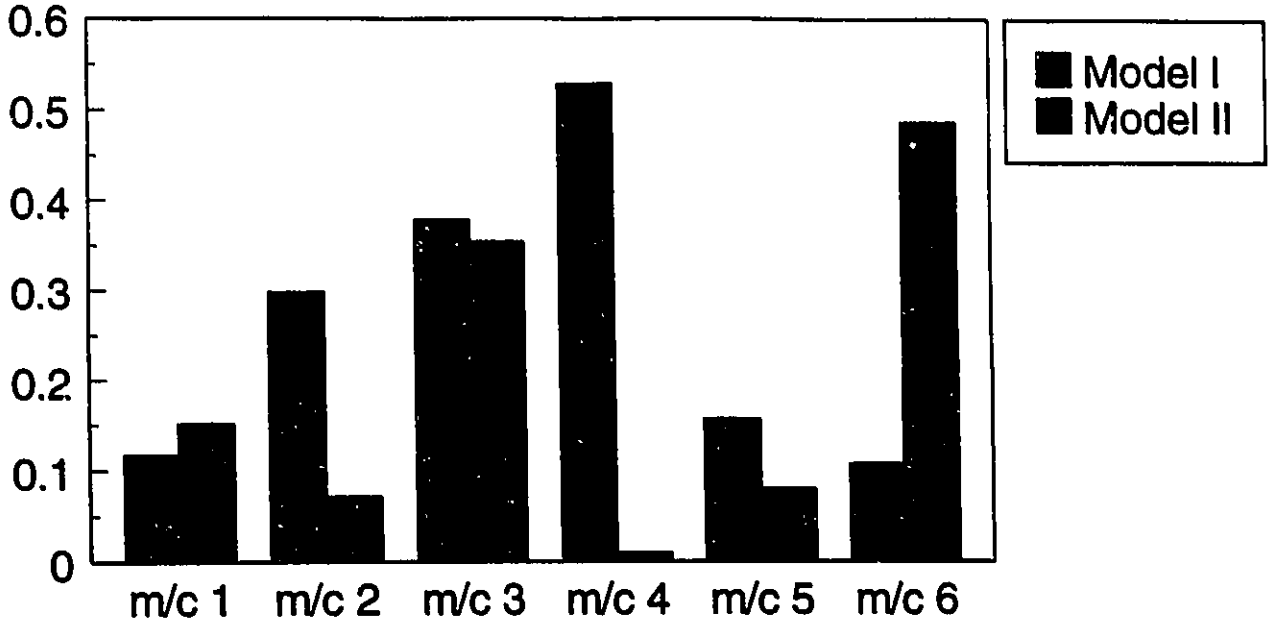


Fig 6.29

Part Type 3

Utilization values

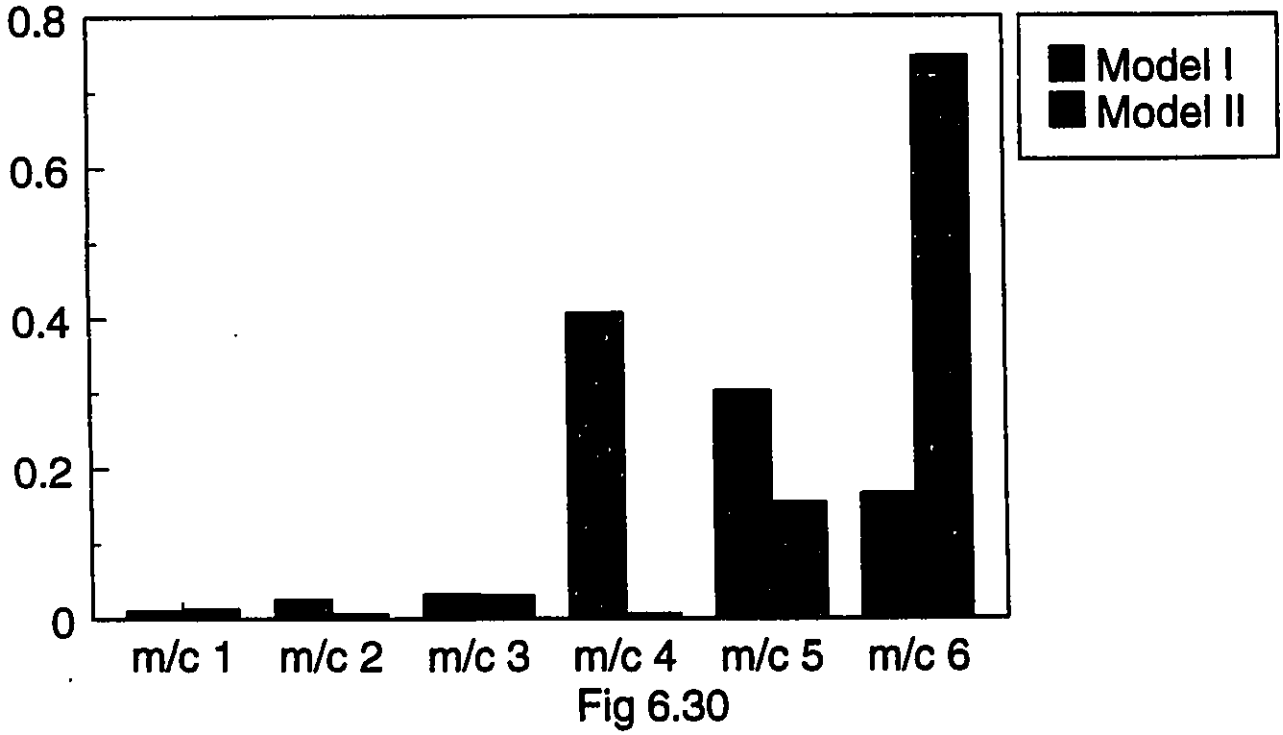


Fig 6.30

Model I and Model III comparison

Part Type 1

Throughput values

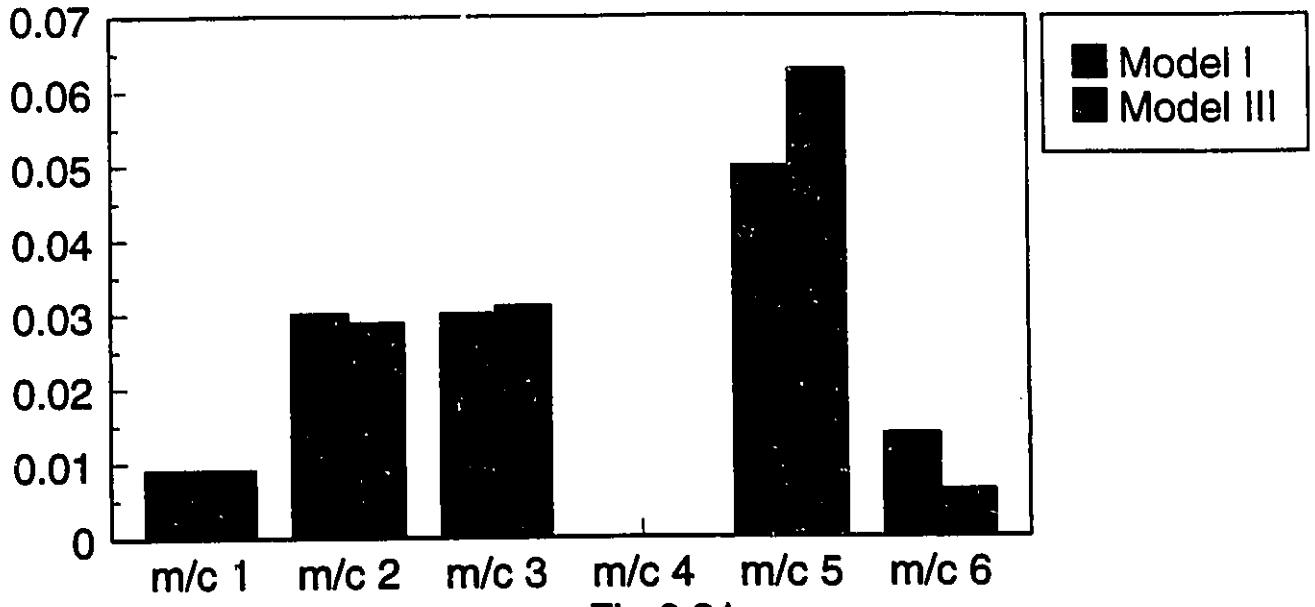


Fig 6.31

Part Type 1

Utilization values

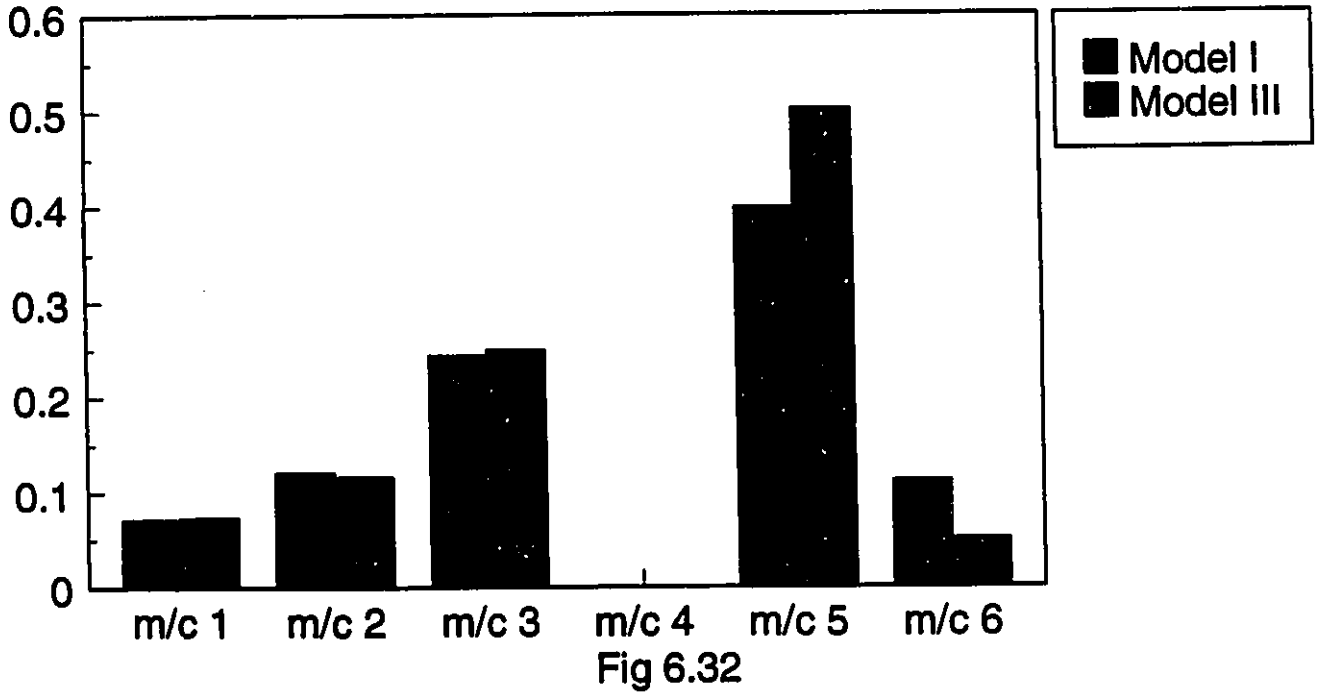


Fig 6.32

Model I and Model III comparison

Part Type 2

Throughput values

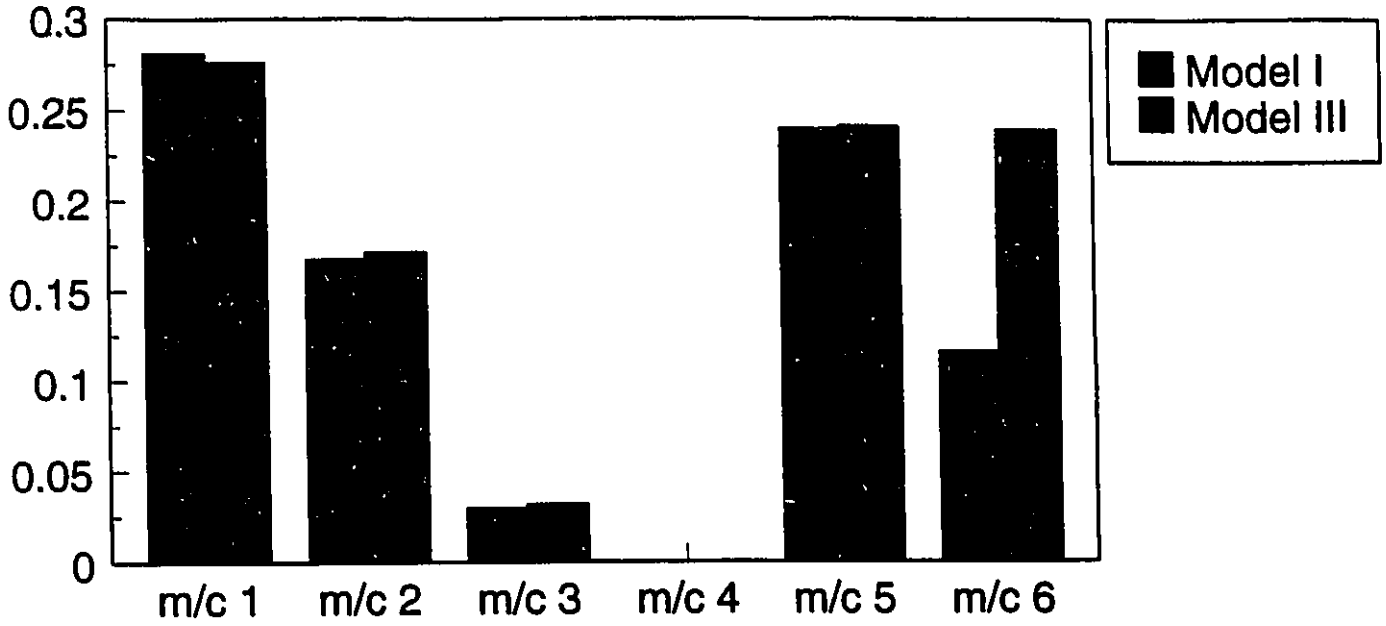


Fig 6.33

Part Type 2

Utilization values

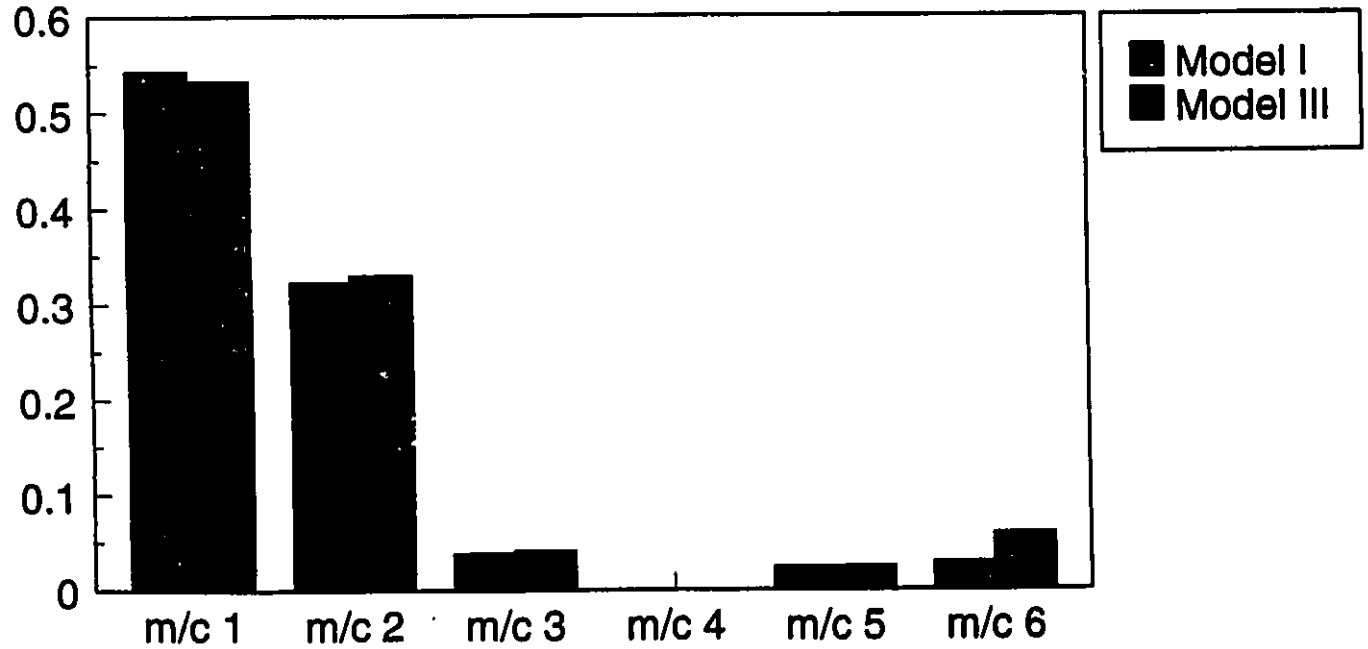


Fig 6.34

Model I and Model III comparison

Part Type 3

Throughput values

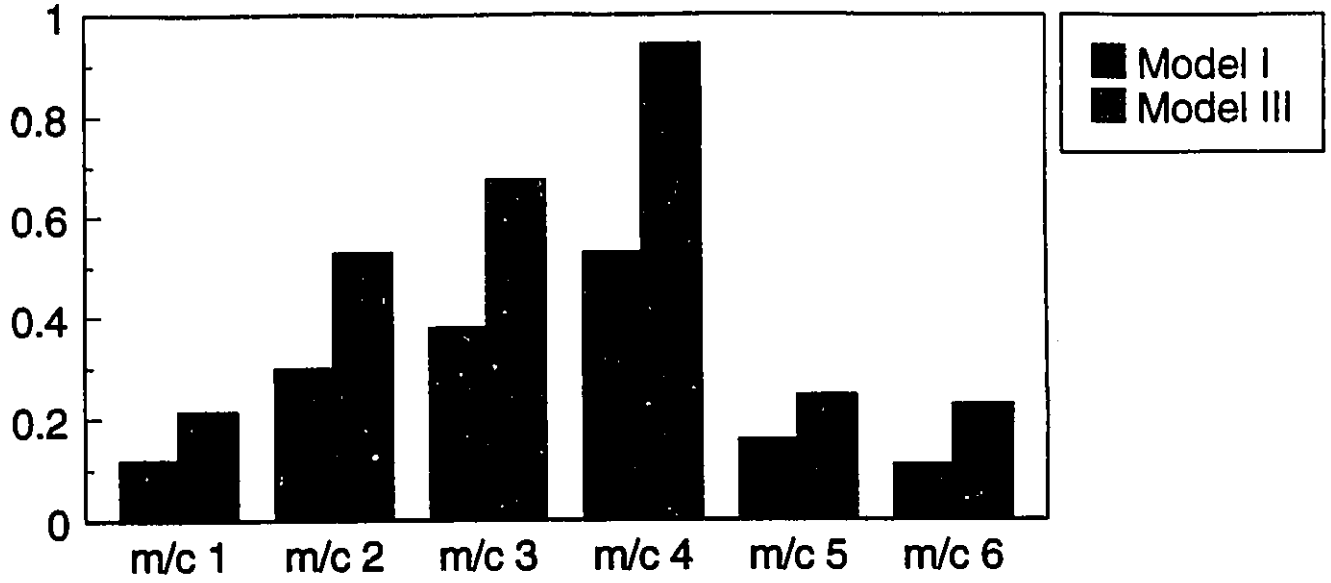


Fig 6.35

Part type 3

Utilization values

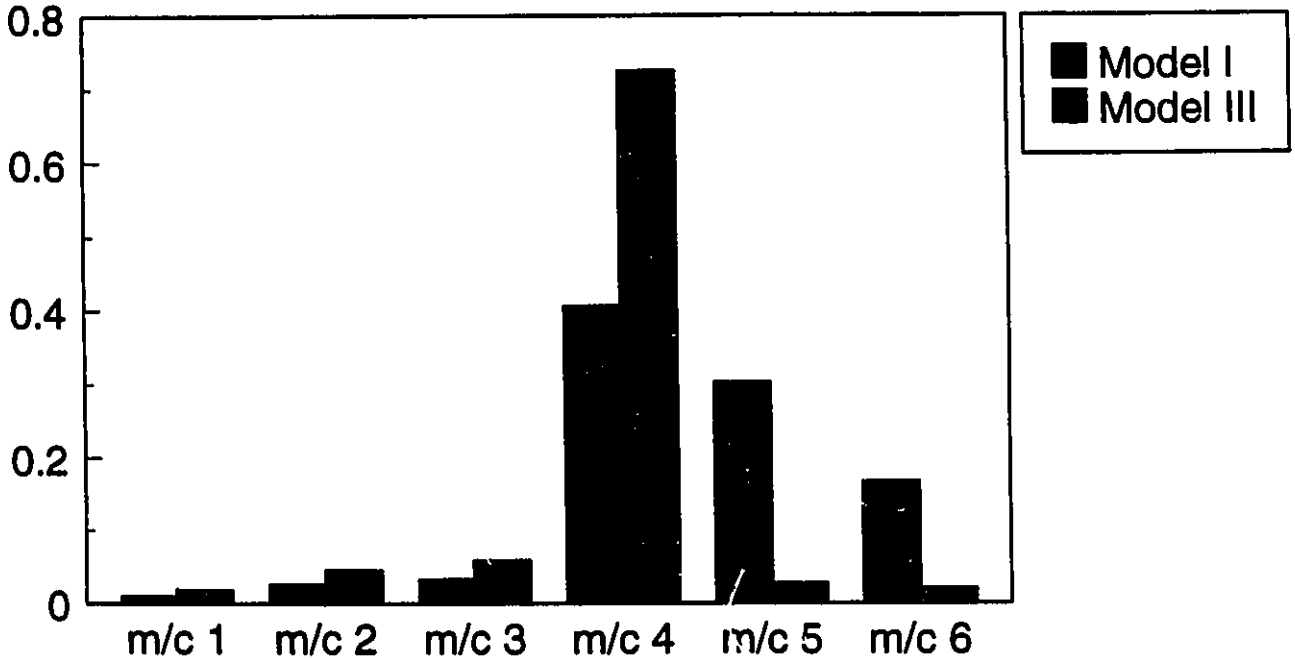


Fig 6.36

Model I and Model IV comparison

Part Type 1

Throughput values

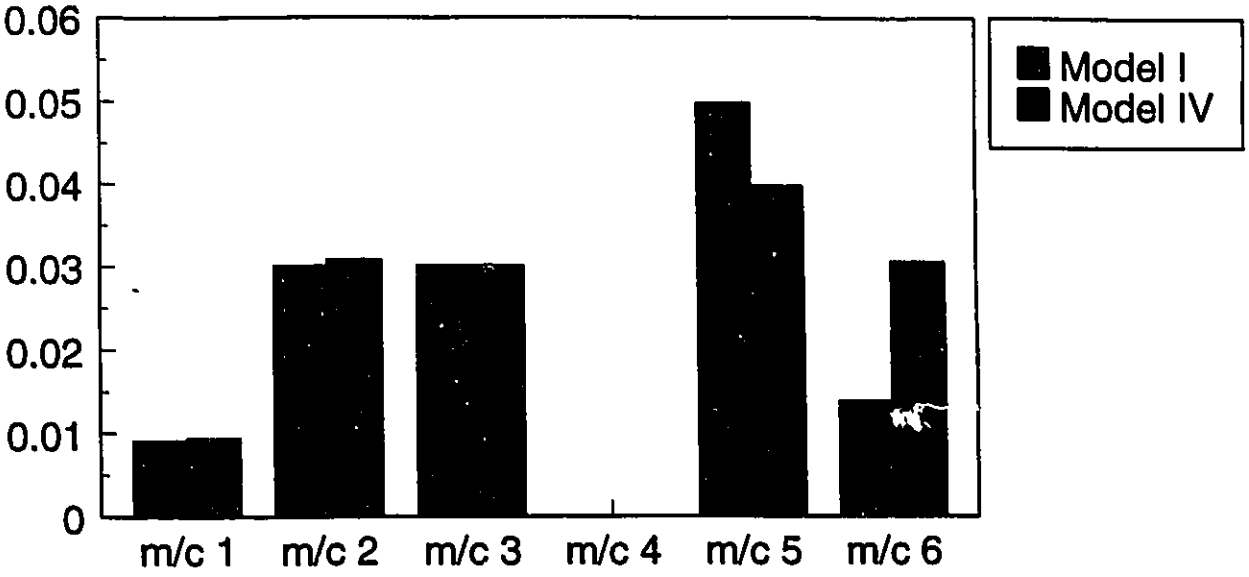


Fig 6.37

Part Type 1

Utilization values

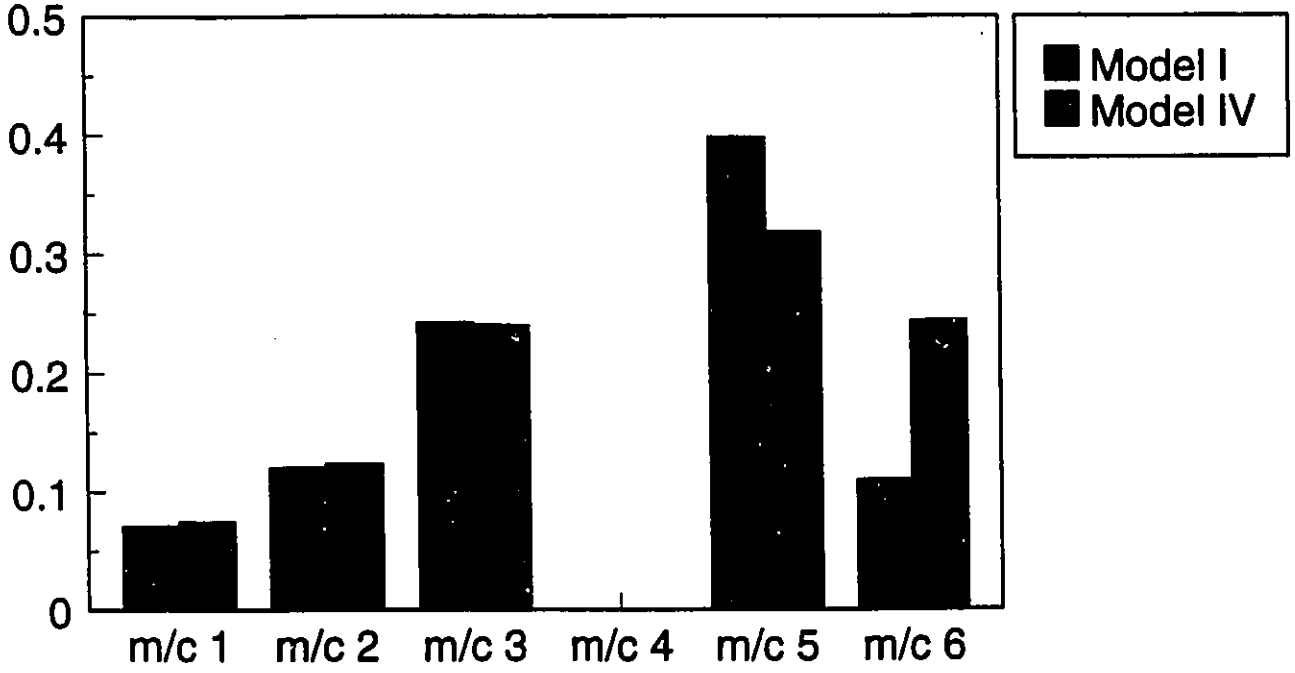


Fig 6.38

Model I and Model IV comparison

Part Type 2

Throughput values

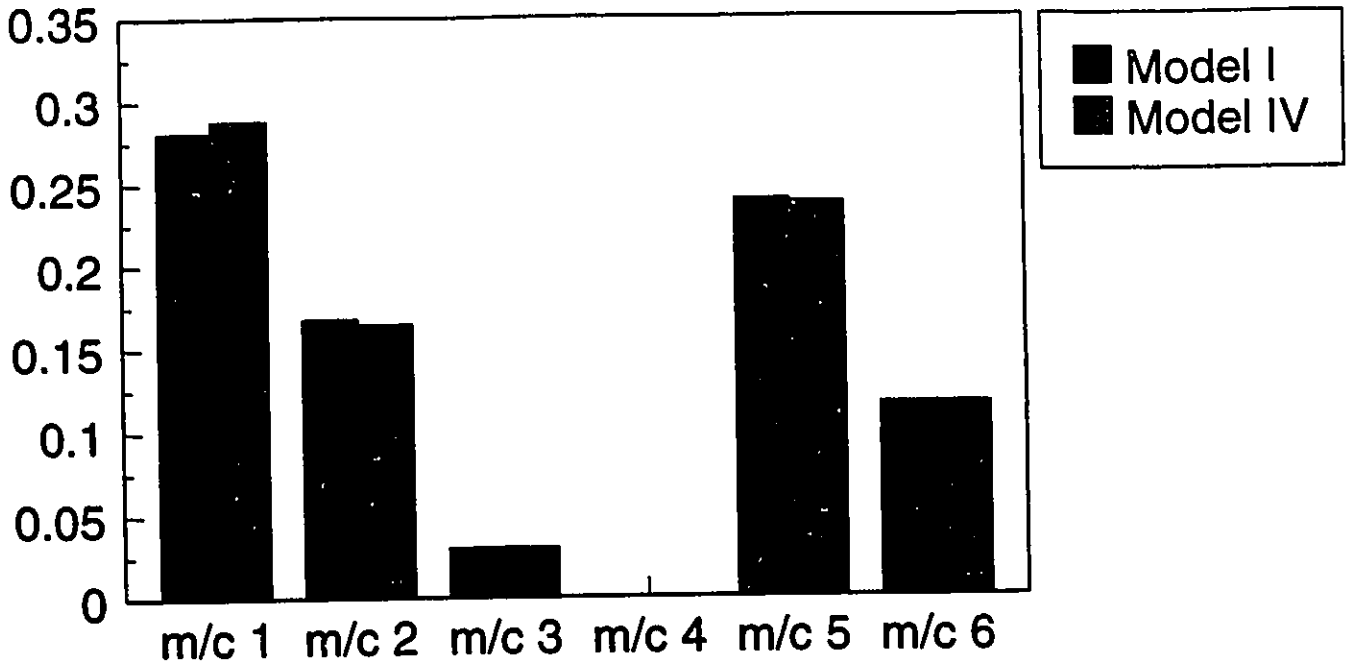


Fig 6.39

Part Type 2

Utilization values

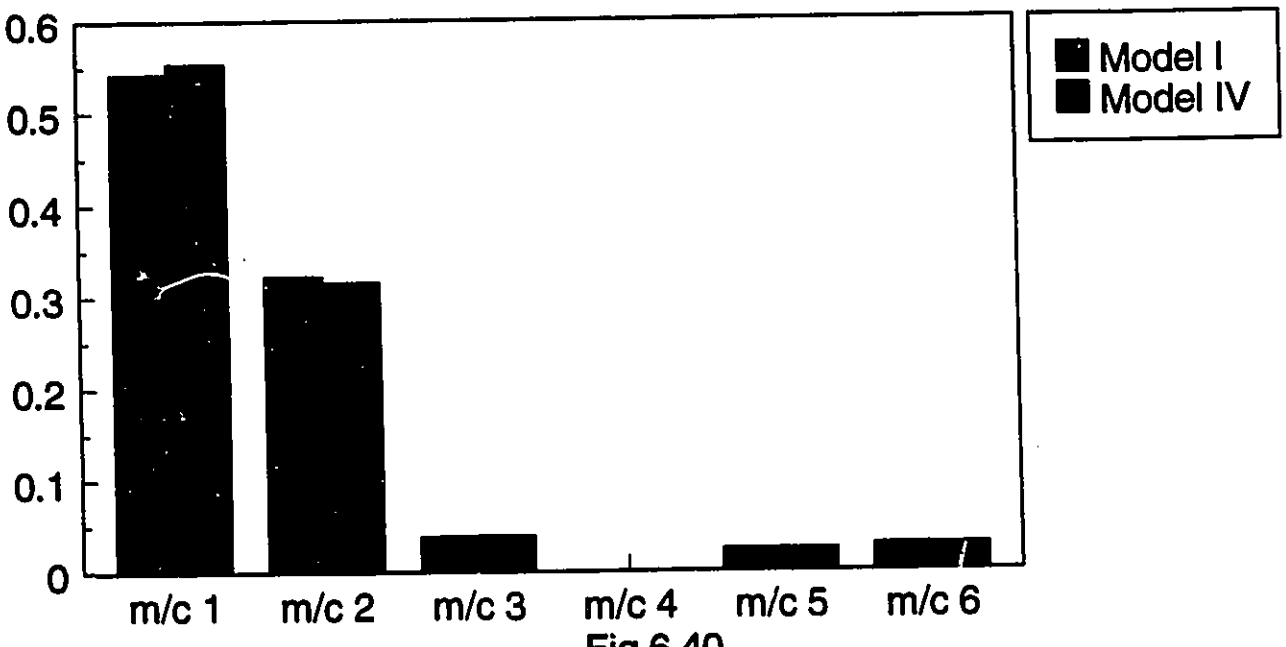


Fig 6.40

Model I and Model IV comparison

Part Type 3

Throughput values

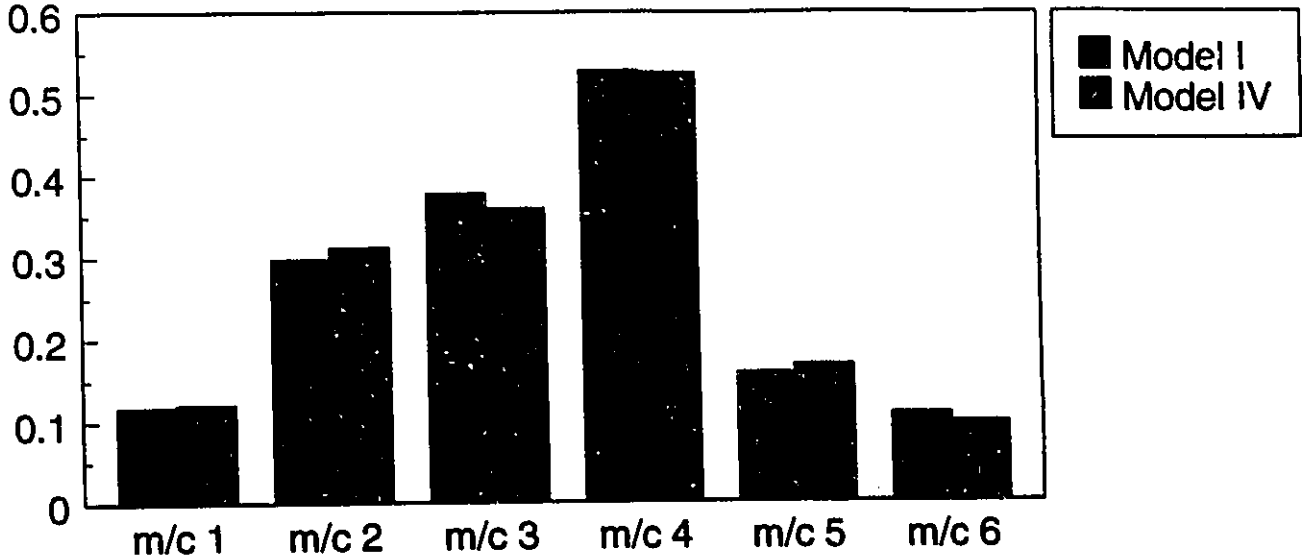


Fig 6.41

Part Type 3

Utilization values

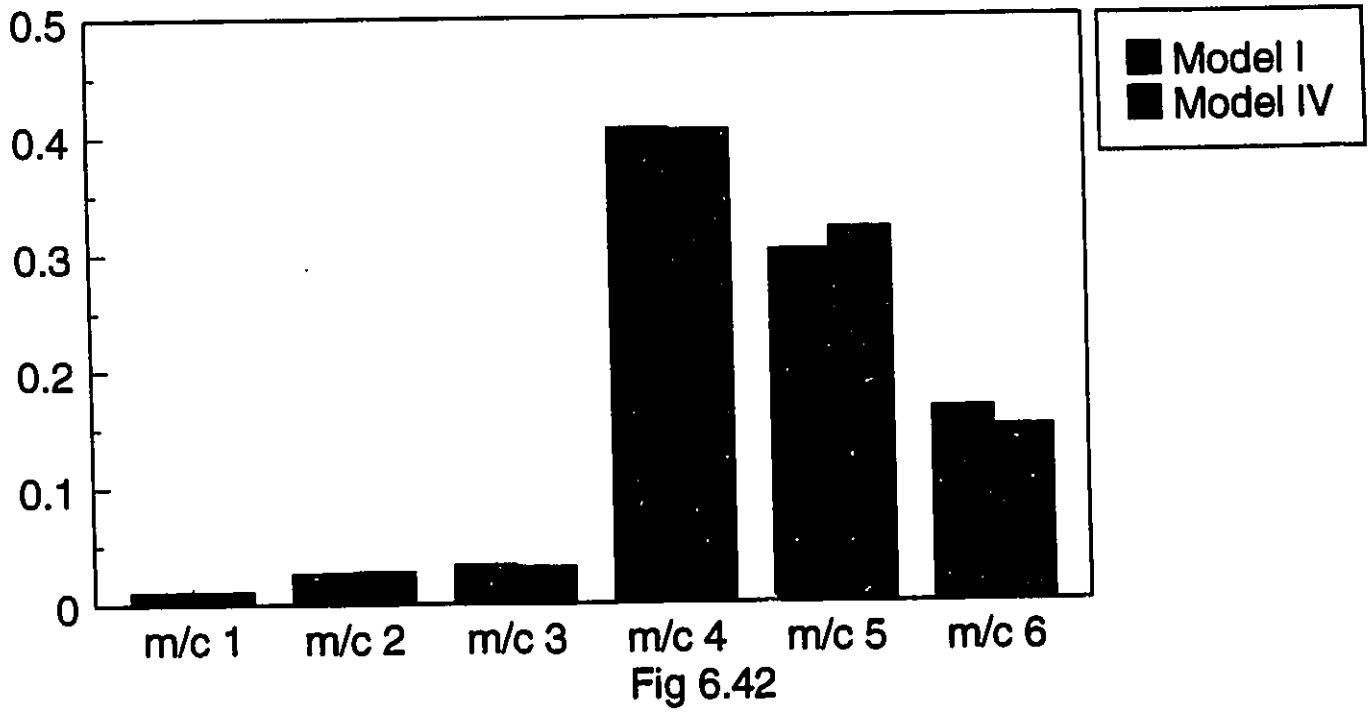


Fig 6.42

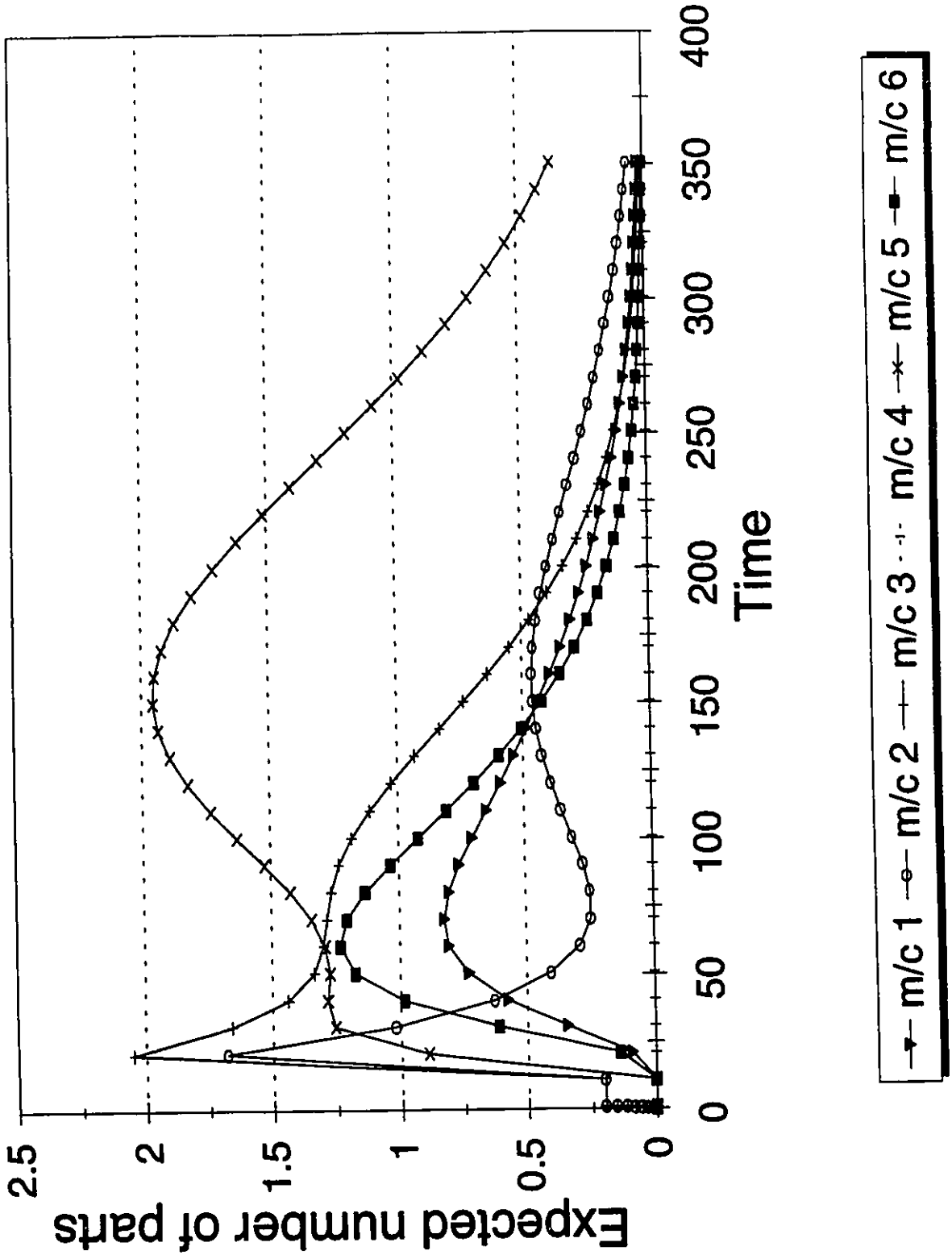


Fig 6.43 Machine loading pattern for Part type 1 in Model I

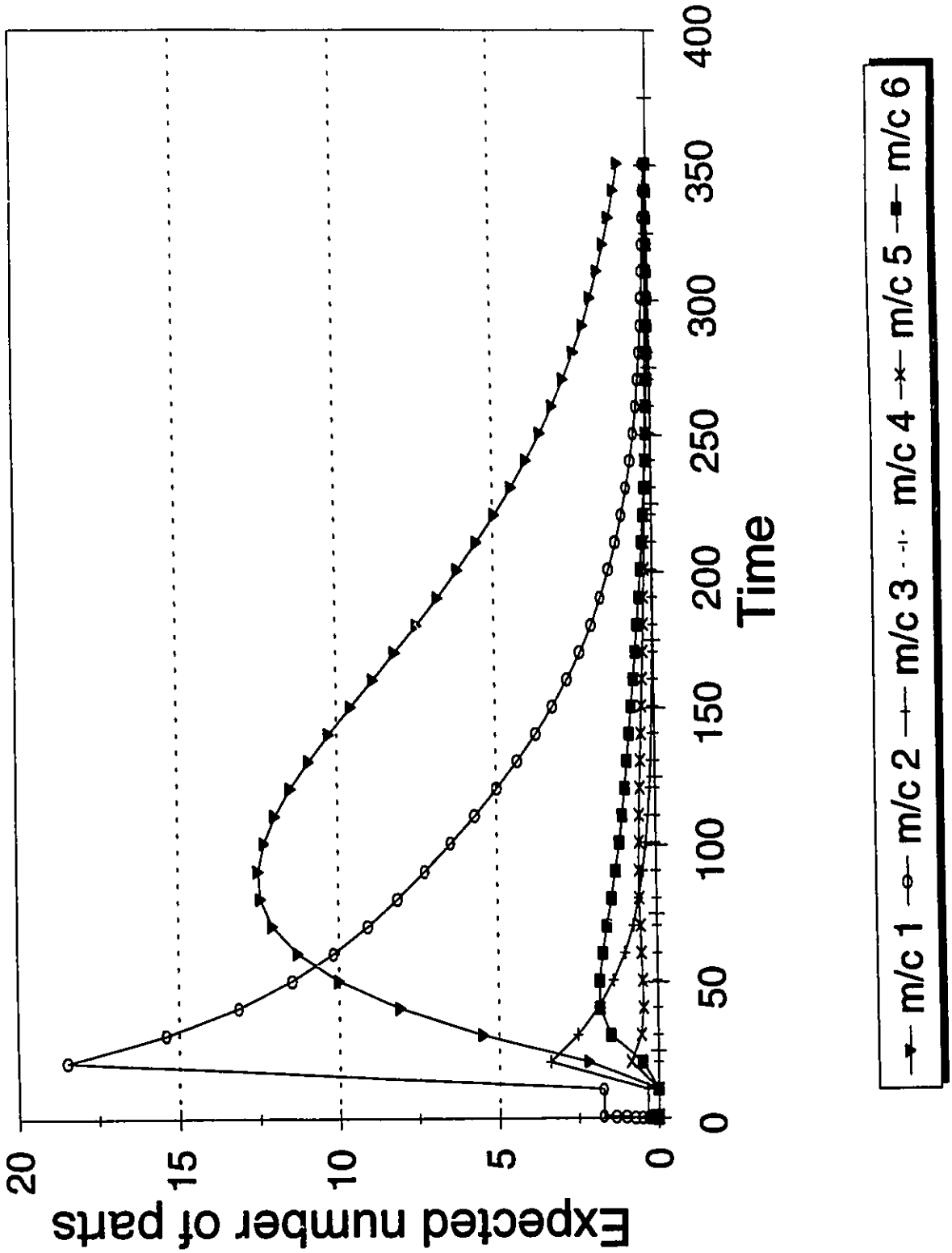


Fig 6.44 Machine loading pattern for Part type 2 in Model I

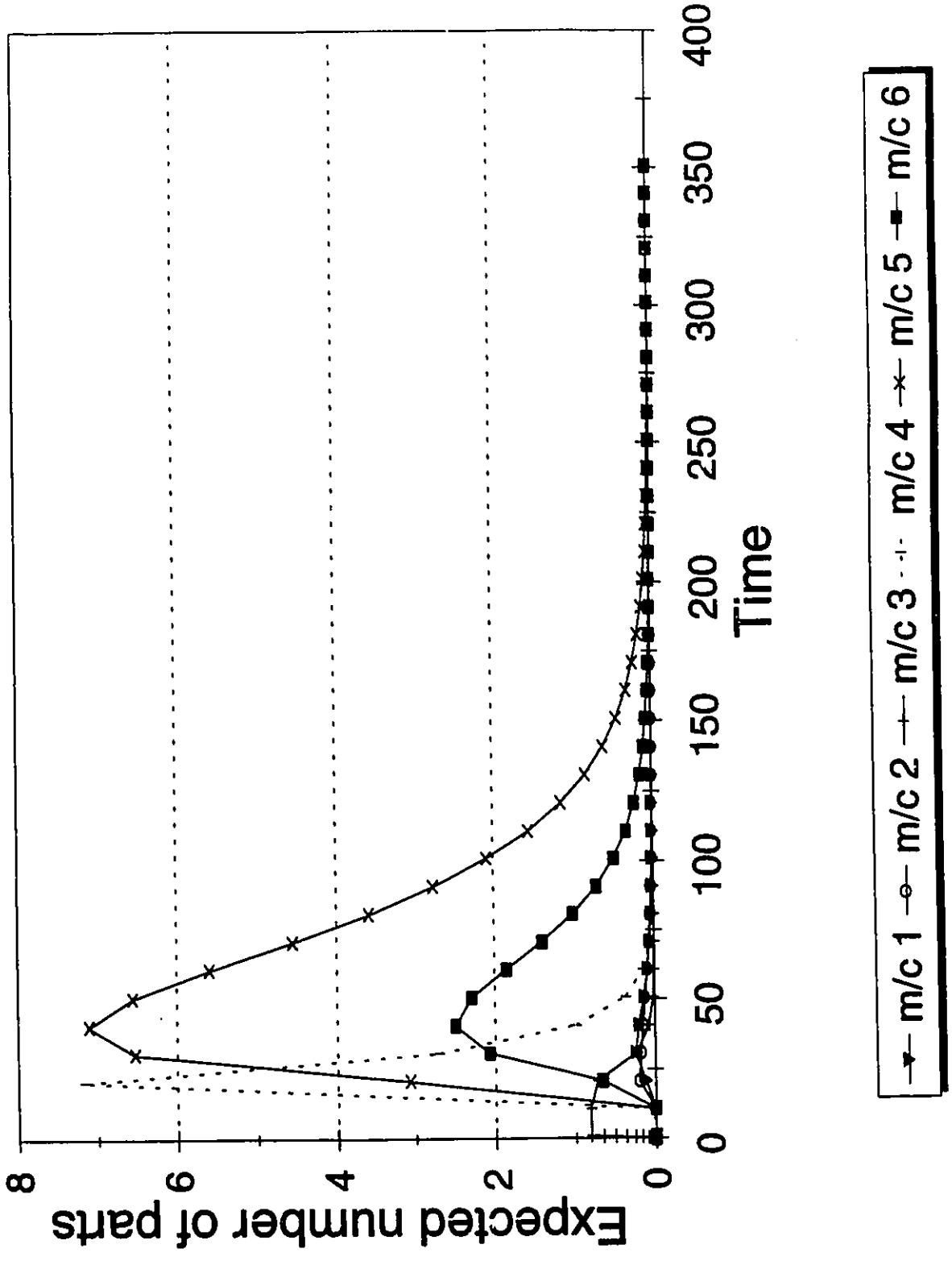


Fig 6.45 Machine loading pattern for Part type 3 in Model I

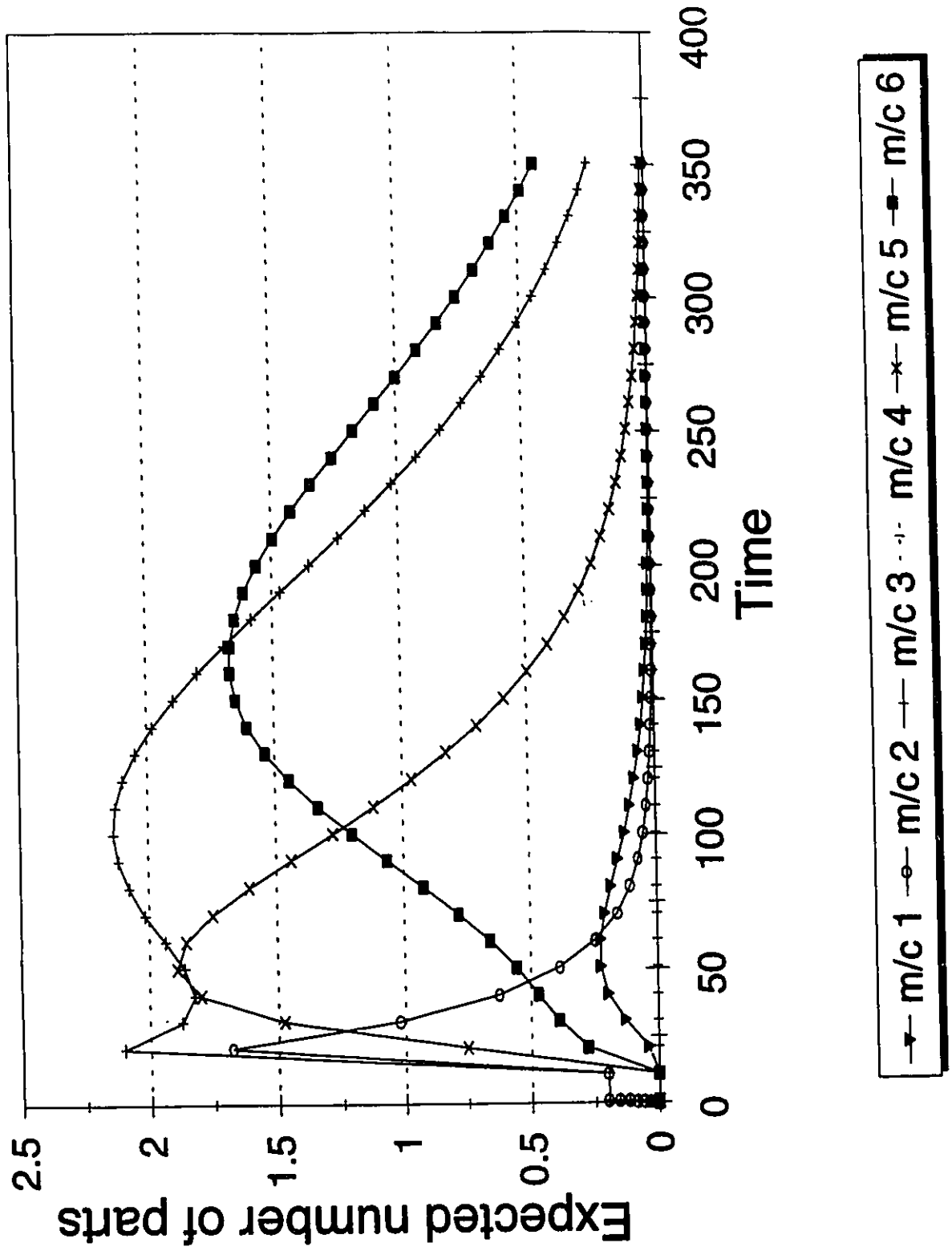


Fig 6.46 Machine loading pattern for Part type 1 in Model II

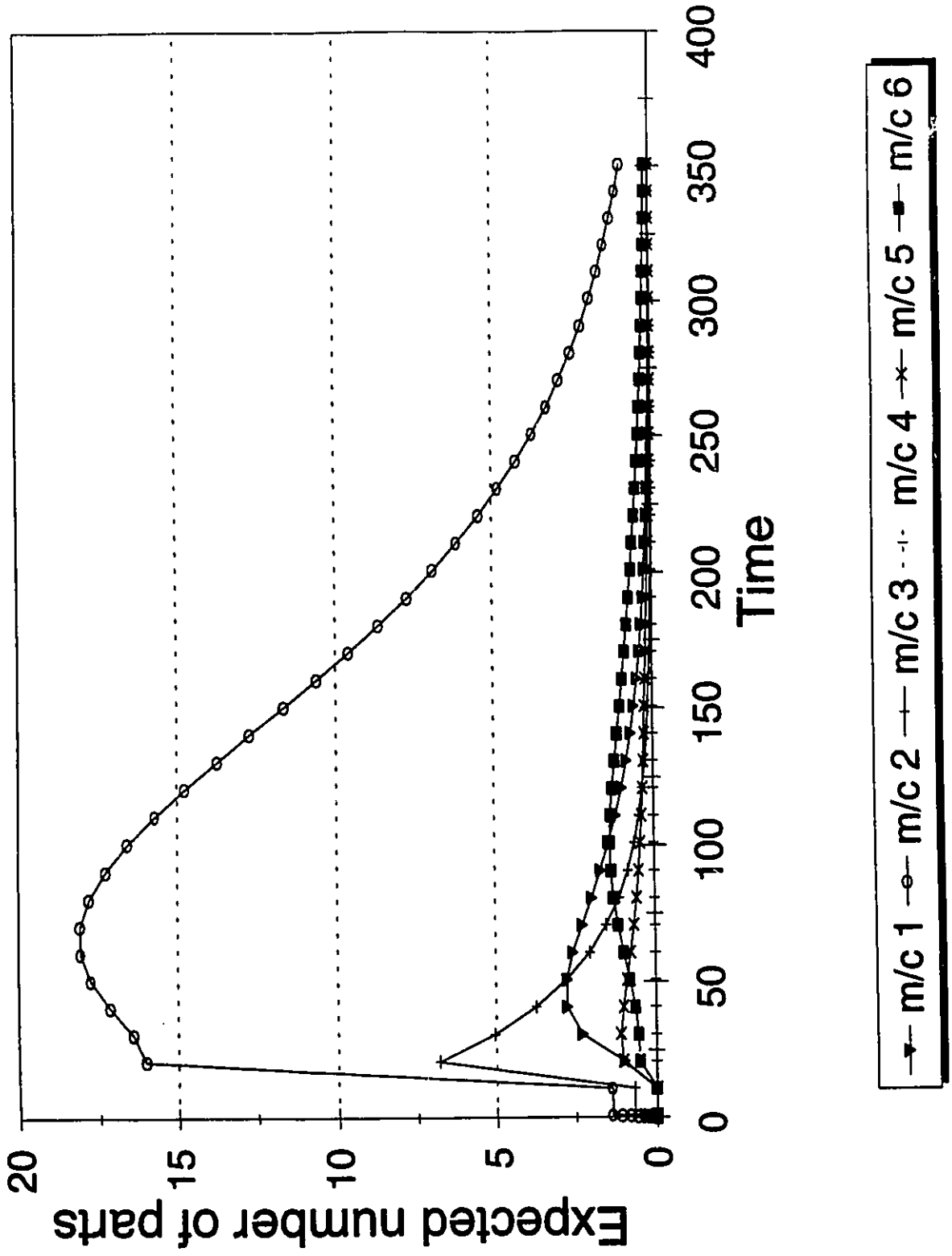


Fig 6.47 Machine loading pattern for Part type 2 in Model II

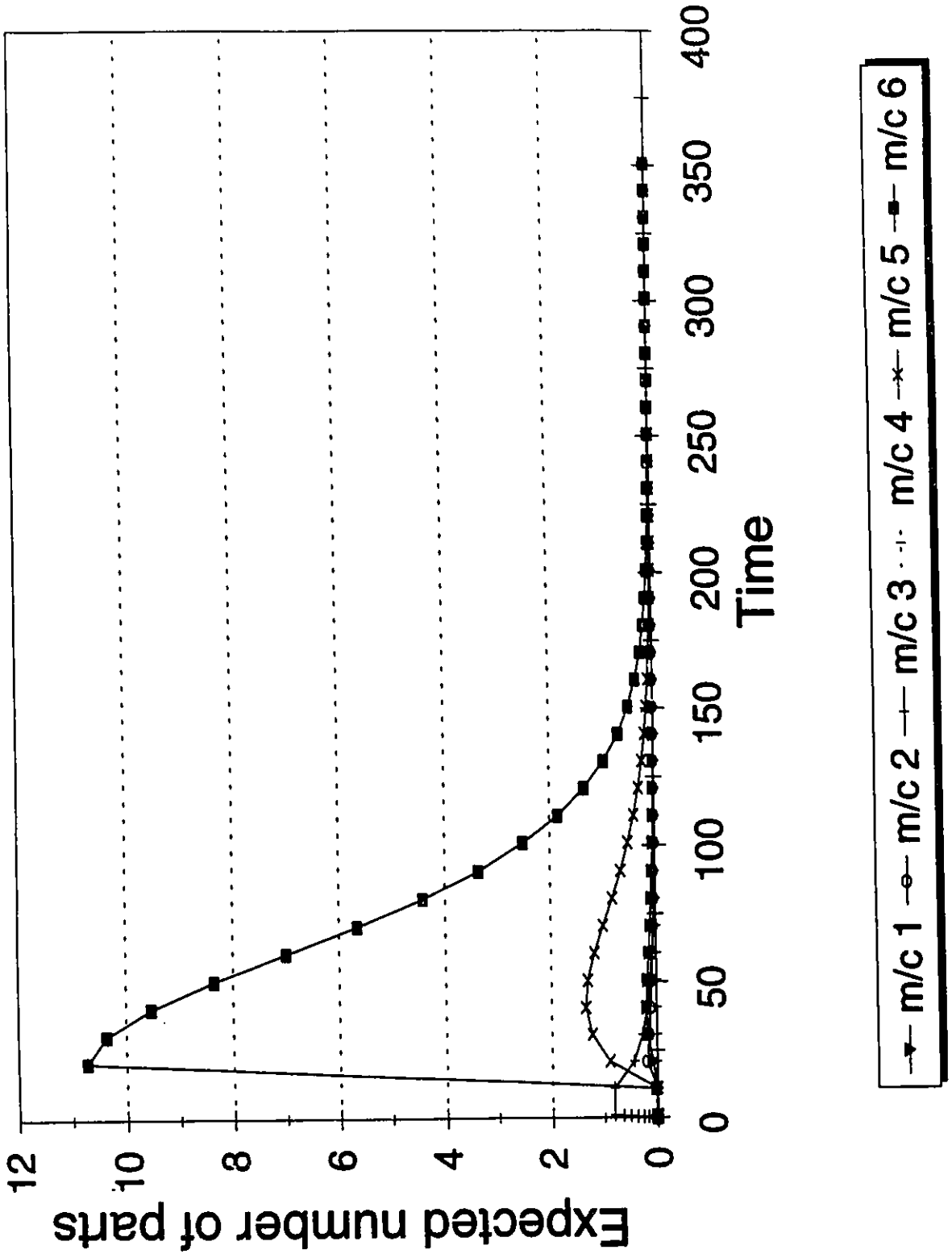


Fig 6.48 Machine loading pattern for Part type 3 in Model II

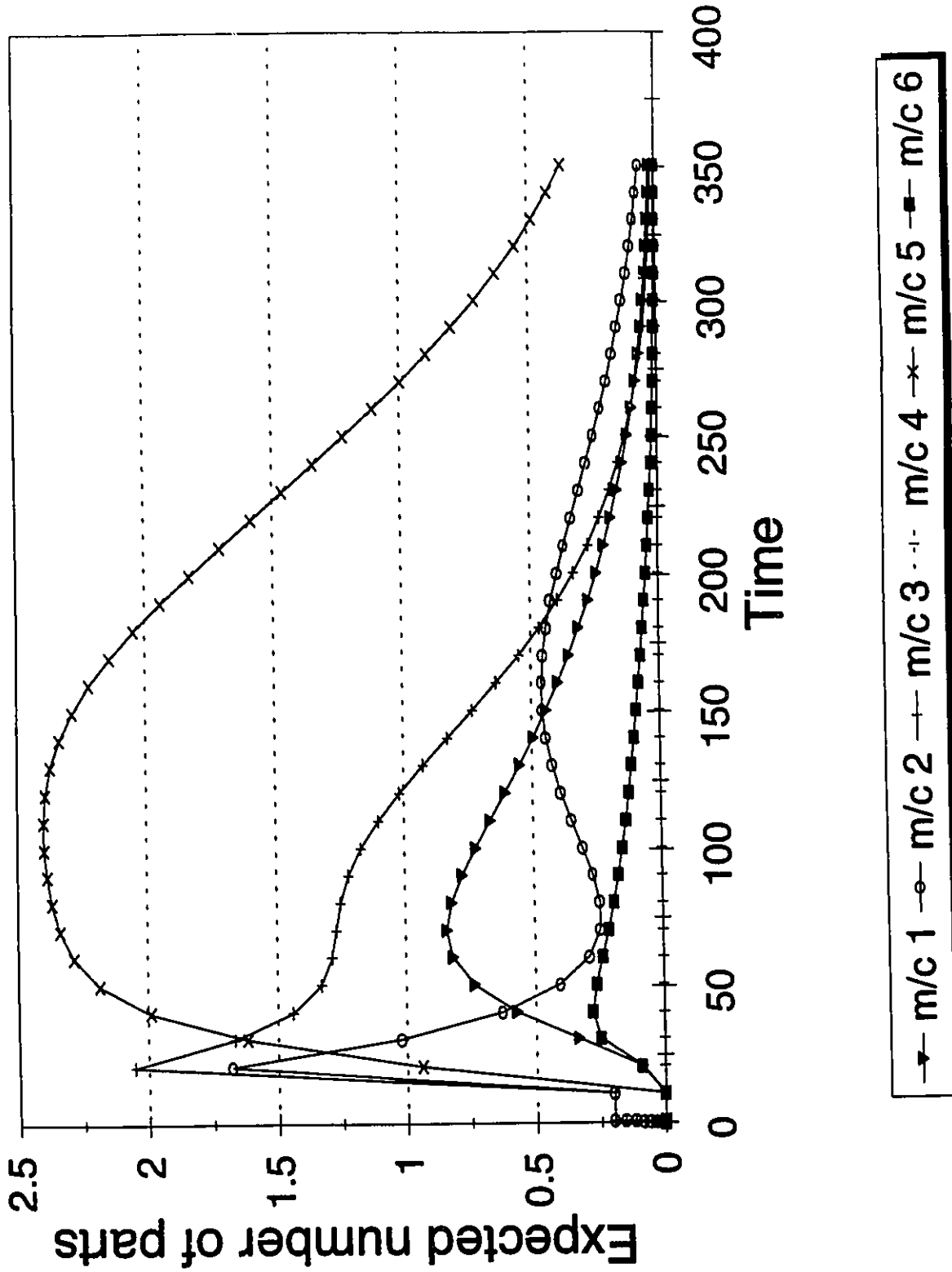


Fig 6.49 Machine loading pattern for Part type 1 in Model III

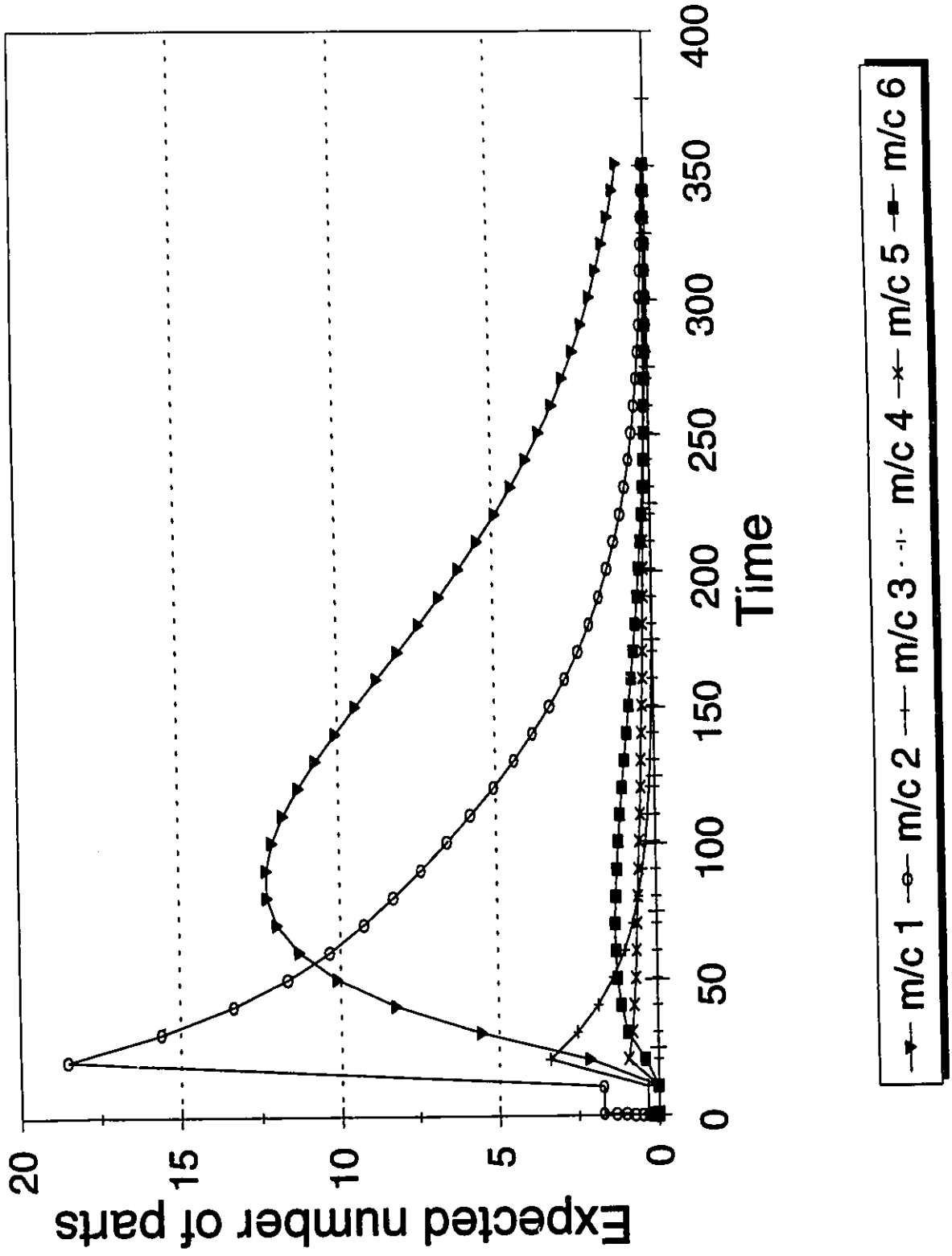


Fig 6.50 Machine loading pattern for Part type 2 in Model III

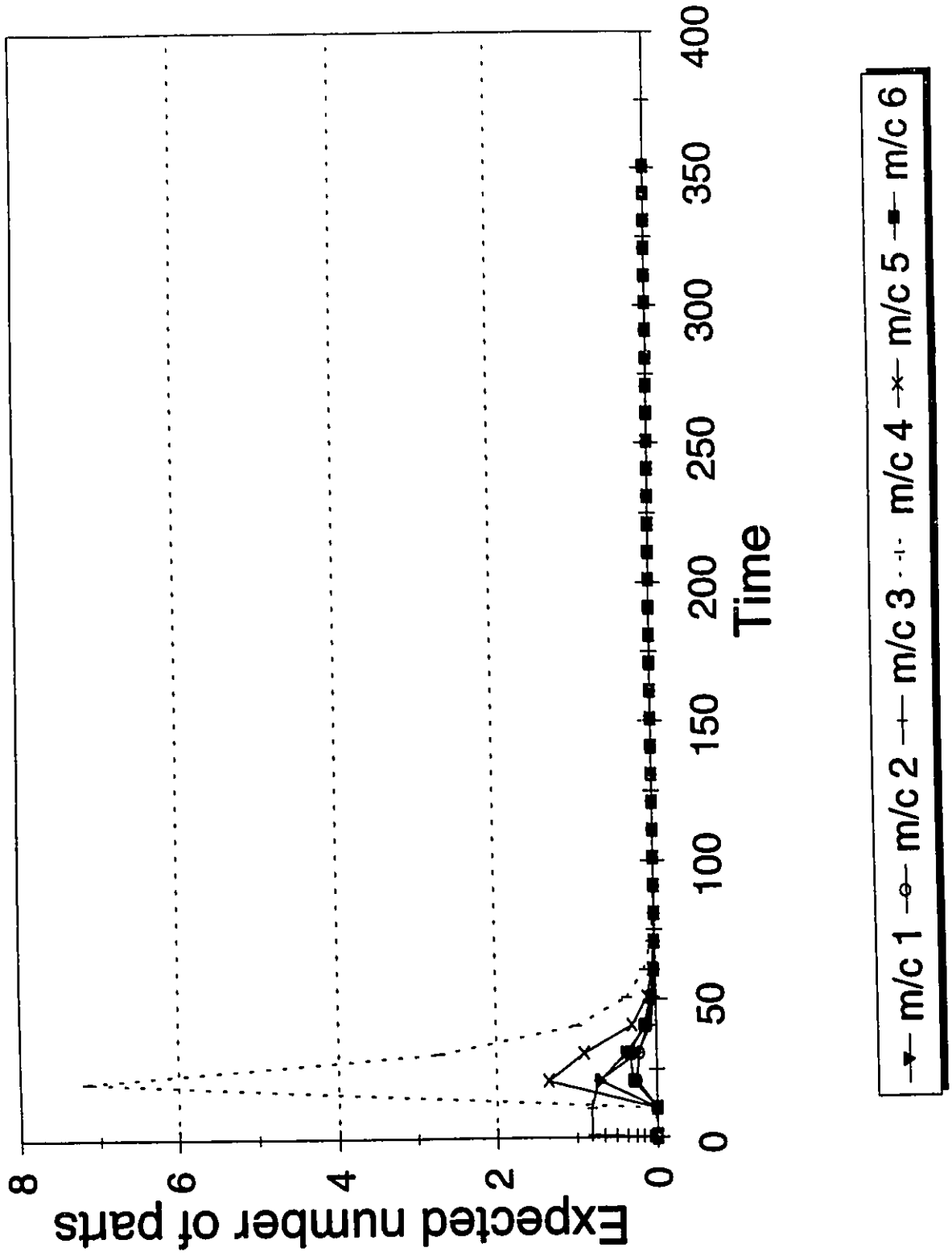


Fig 6.51 Machine loading pattern for Part type 3 in Model III

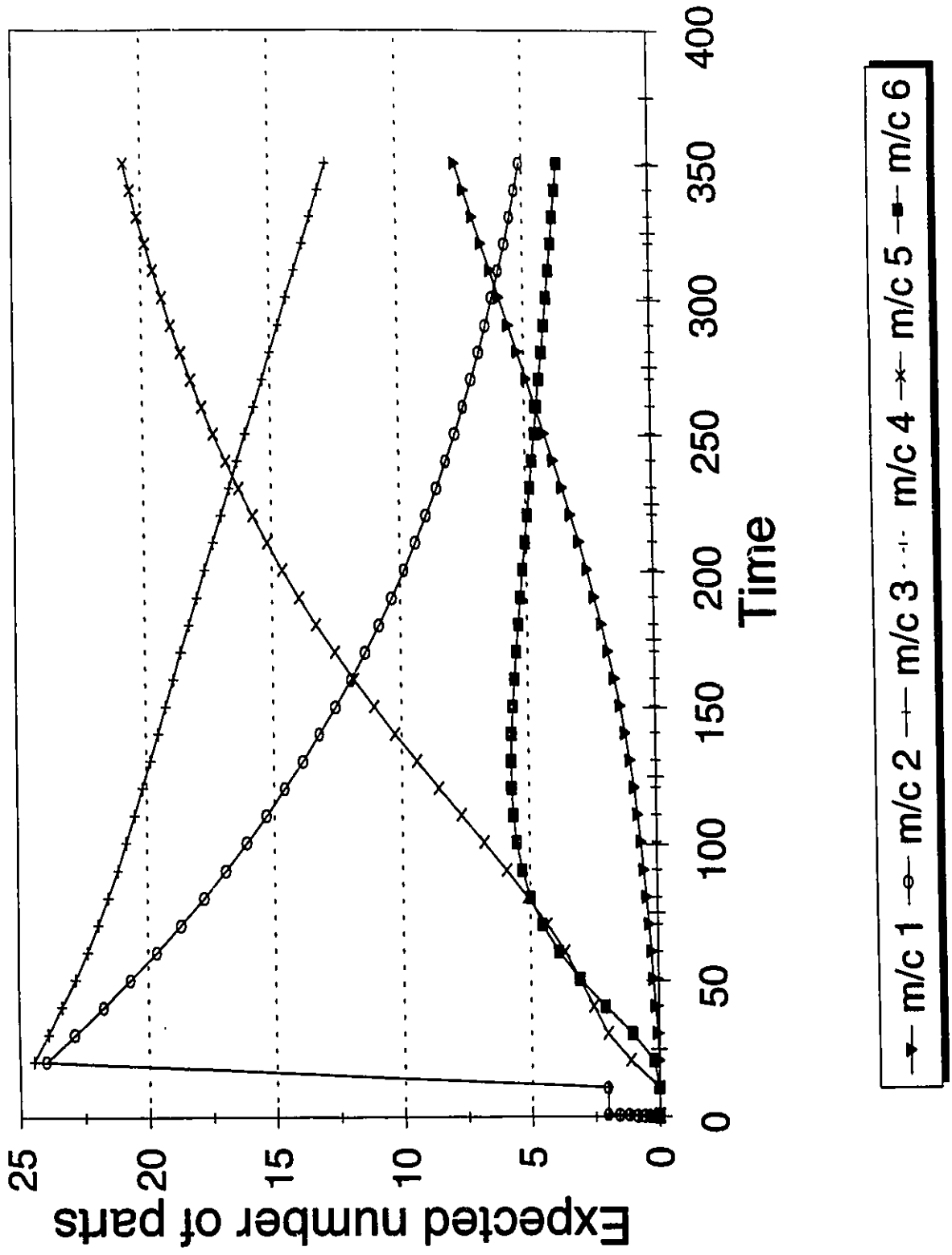


Fig 6.52 Machine loading pattern for Part type 1 in Model IV

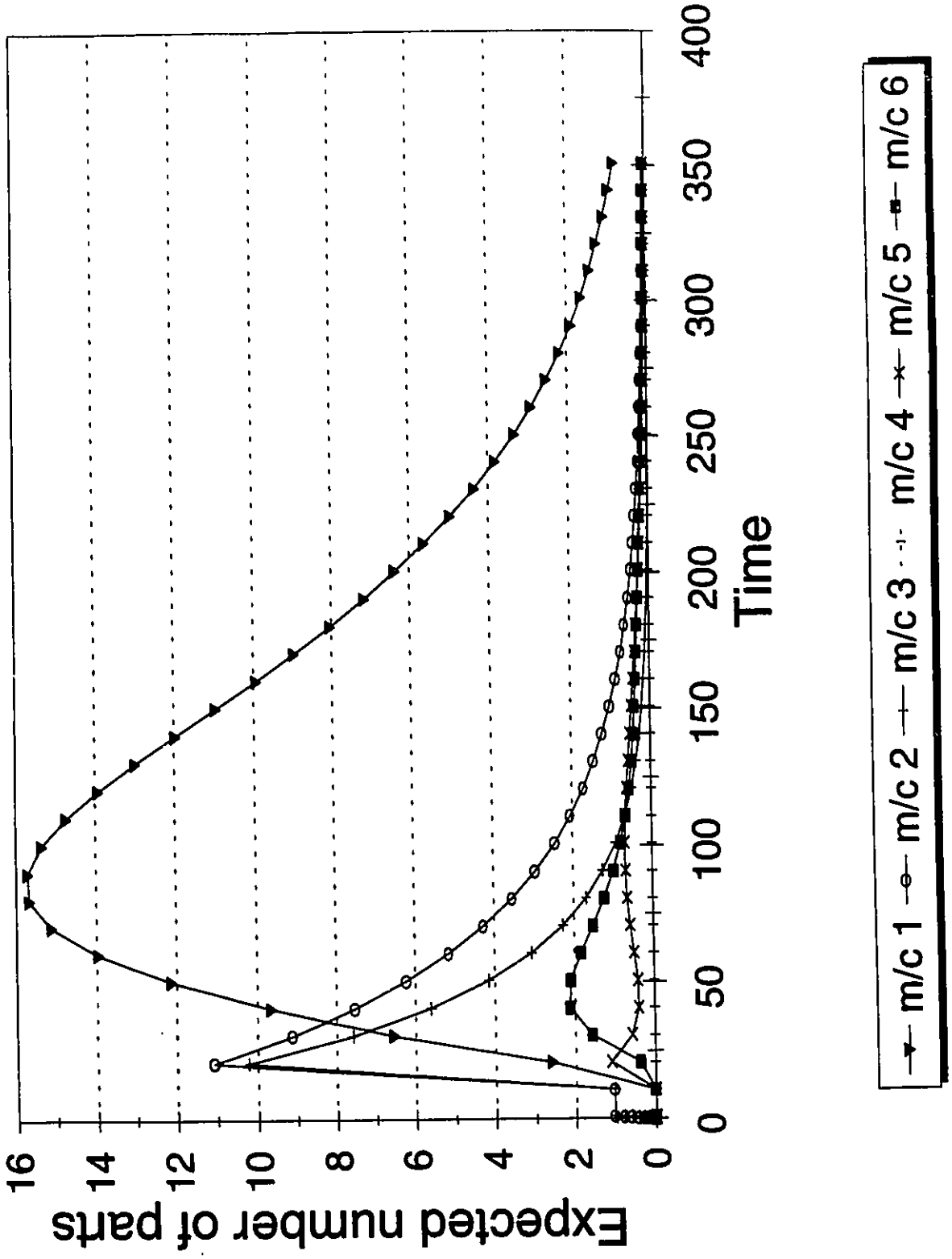


Fig 6.53 Machine loading pattern for Part type 2 in Model IV

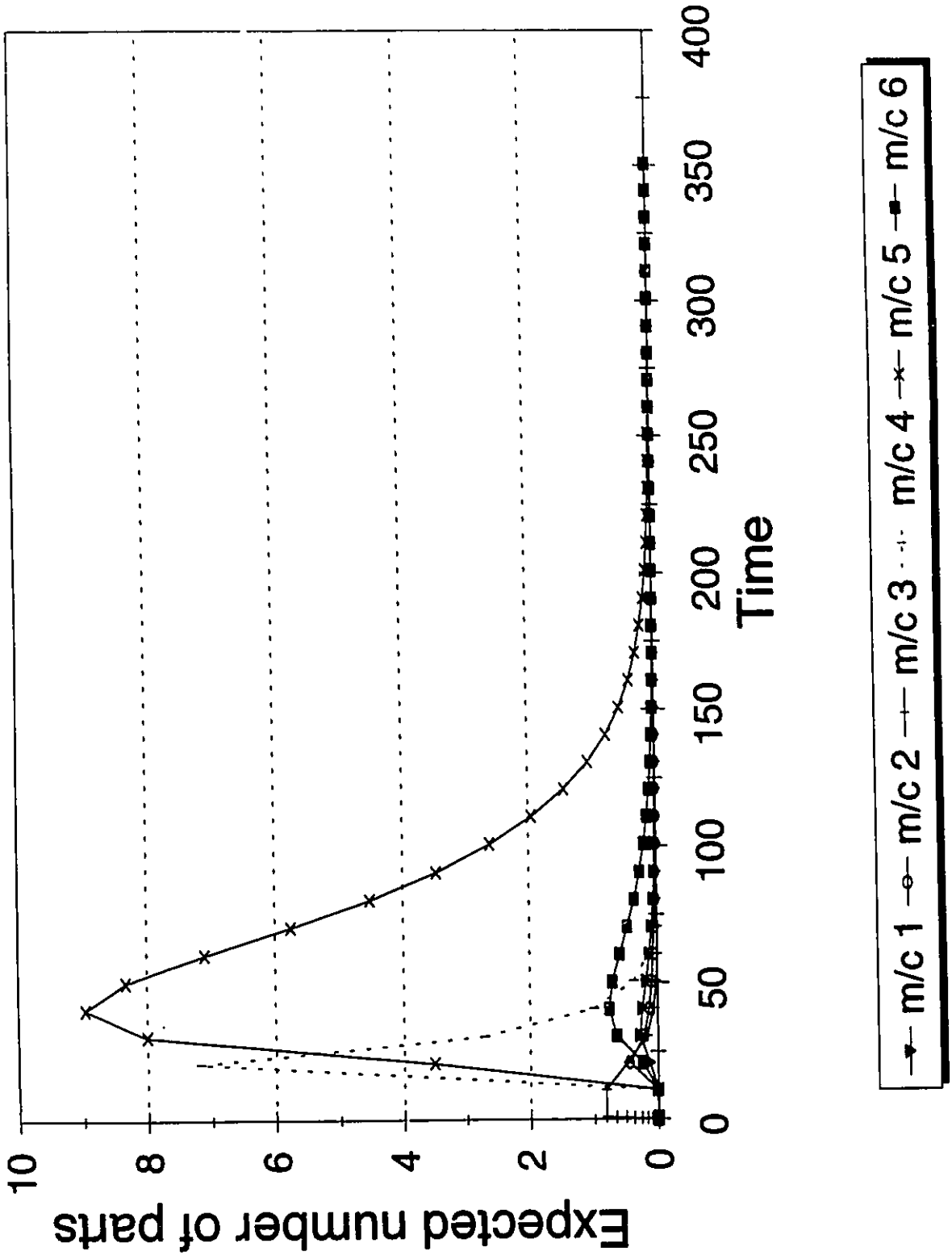


Fig 6.54 Machine loading pattern for Part type 3 in Model IV

Chapter 7

DISCUSSION AND CONCLUSION

7.1 About the Petri Net Model

The Petri Net model developed in Chapter 5 is a scaled down representation of a real life problem. It handles 3 part types and about 90 parts being routed through 6 machining centres. It should be noted at this stage that one of the constraints in the process of working with Petri Nets is the enormous state space generated. An effort has been made to keep the state space within a reasonable size (13000 states) in order to stay within the computational capacities of the software and the hardware being used. Nevertheless, care has been taken to represent the critical and significant components of an auction based manufacturing system. Caution has been exercised to ensure that certain features balance out the place transition diagram so that instances of too much detail on less significant features are reduced and the more significant features are highlighted. The study demonstrates a technique for controlling as well as predicting the expected behaviour of an auction based manufacturing system.

7.2 Effectiveness of the steady state and transient analysis

An auction based manufacturing system is a discrete event dynamic system (DEDS) which can go through a variety of states and events. To understand the behaviour of such a DEDS it is necessary to study the system changes both through a transient time scaled analysis and a long term steady state analysis. The transient analysis would reveal the immediate consequences in a system as a result of tool non-availability or breakages, machine breakdowns, deadlocks and change of part mixes while a steady state analysis is more likely to give an overall picture of the system performance in terms of its total improvement or deterioration. The transient analysis results discussed here (See Figs. 6.1-6.24 and Figs. 6.43-6.54) show such a variation in one of the system parameters (i.e.the expected queue length at a machining centre). Each line representing the expected queue length at a machine represents the expected pattern in which it would be loaded corresponding to a time scale. Since the transient analysis gives probabilistic or expected values, it is necessary that to be meaningfully utilized for making control decisions, one has to fix a threshold limit. If, at a particular time instant, a machine is expected to be loaded above this threshold then it should be interpreted as a busy machine and the corresponding associated resources ought to be made available to it. If, on the other hand the same machine is found to be loaded below this threshold, then the resources can be withdrawn. While this can help system management significantly by leading to better utilization of resources, it must be kept in mind that after all it is an

expected trend and need not always be that way. Hence the merit of decisions taken after such an analysis would depend on the threshold value chosen, which is crucial to deciding the probability of making successful predictions about machine loading patterns.

The steady state analysis yields parameters representing the system behaviour like machine utilization and throughput values on a long term basis. This is what the machine parameters would be if the system continued running in a similar fashion until a hypothetically infinite amount of time. Of course, not all systems are deadlock free, hence it is improper to rely solely on the long term steady state trends of a DEDS. Any manufacturing system however well maintained and organized, faces stoppages due to breakdowns and repair which might reflect themselves as deadlocks in the corresponding Petri Net model representation of it. A steady state analysis cannot always be done on a system with deadlocks (Viswanadham and Narahari, 1992). Moreover, in today's highly competitive manufacturing scenario, it is more of interest to track the time based system variation on a real-time basis and stream line expensive resource allocation so that there is as little wastage as possible. This calls for accurate information on the demand for resources on a time scale. Hence in conclusion, it is necessary that a complex DEDS like an auction based manufacturing system be understood through both steady state and transient analysis methods.

7.3 Computational Experience

One of the biggest constraints while working with Petri Nets is the size of the state space generated. It is imperative to strike a compromise between a computationally feasible state space and a minutely detailed model of the system. It is possible to model the characteristic and most significant features of a system well without going into minute details which might otherwise lead to unnecessary increase in the state space.

The model studied here has three subnets each of which represents the events related to a particular part type. For every additional part type an additional subnet will have to be added to this model. This is due to the limitations of this type of Petri Net because of which it is not possible to identify or differentiate between tokens from different subnets. One of the most efficient ways to tackle this problem would be through the use of "Coloured Petri Nets", (Martinez et al., 1986), (Viswanadham et al., 1987). In this kind of Petri Nets, the tokens have an associated colour or identity and hence can be differentiated easily from each other. This approach is very powerful and useful in reducing all identical subnets to a single subnet. Thus it prevents duplication of subnets and helps in shrinking large Petri Net representations to smaller and simplified forms.

The models studied here have a maximum state space of 13000 states. It takes about 6 minutes for the steady state analysis of each model while the transient analysis takes

about 180 minutes to generate the data for a time scale from 0 to 350 time units with intervals of 5. All computations have been performed on a Sun Sparc station with 32 MB RAM. The Petri Net solver is SPNP ver 3.1, developed at Duke University, North Carolina.

7.4 Major objectives that have been achieved by this work

One of the principal objectives of this work was to develop a robust and realistic model of an auction based manufacturing system. Analysis of such a model was expected to give the user a set of predictions or trends that are most crucial to utilizing a heterarchical auction based manufacturing system. It is felt that this knowledge base could be used in conjunction with the results of (Zhou,1993) and (Upton et al.,1991) to better understand an auction based manufacturing system.

The models developed are robust and compact and have the following improvements over the previous work (Zhou,1993).

1) The model handles tokens representing unique part types. While only three part types have been studied here owing to computational constraints, the prototype developed can handle any number of parts as long as the state space is manageable.

2) The concept of process plans has been integrated into the model. Each part type has

a process plan which can have any number of stages in it. Also each stage of a process plan can have more than one machine available as alternatives for machining that particular stage. This addition makes the model more realistic and allows the operator to suggest secondary and tertiary machining centres capable of performing the same task.

3) The concept of multiple parts within each part type has been integrated into the model. This makes the model more realistic and also helps in performing a more complete study on the sensitivity of the system behaviour to variation in the number of parts i.e. the part mix.

4) The behaviour of the models have been studied for the cases of different part allocation heuristics and tie-breaking rules. Also the model behaviour to variation in the machining rates of the machining centres has been studied.

5) The models have been analyzed through both transient time and steady state analysis techniques. Methods have been suggested to demonstrate how information available from both these techniques can be integrated together to draw conclusions about expected system behaviour.

7.5 Proposed future extensions

While the operator has to manually change the different control parameters (like m/c rate, number of parts, control heuristics) to study changes in the response of the system, it would be ideal if the system itself could explore these various options automatically and come with a suggestion on the most optimal combination of number of parts, part types and control algorithms. This information could then be stored in an information database so that in the future an expert system could pick the best combination of control algorithms, part types and parts for new scenarios based on already available information.

Also the set of machines to which bids are sent in the study was fixed and limited to six. An extension of this work could explore the possibility of having a much larger number of machines that are grouped together into "virtual cells" with an objective of optimising the transport and machining costs. Different groups of such "virtual cells" would have to be explored to evaluate and extract trends that indicate the best possible combination of machines for a particular combination of parts, part types and control heuristics. Information regarding these trends could be stored as data files which are subsequently accessed by an expert system engine that acts as a decision support aid for quick suggestions during day to day operations of an advanced auction based manufacturing factory.

The study done here is based on Stochastic Petri Nets, but only two types of transitions could be taken into account. A transition can be either immediate or exponentially distributed. This is both due to the limitations of the software used as well as the lack of good and converging numerical methods for performing the steady state and transient analysis of CTMC (Continous Time Markov Chains) for different kinds of firing distributions. As an extension to this work, modelling the same system under more variety of probability distributions could be considered.

7.6 Conclusion

The work presented here demonstrates a technique to improve the understanding of the behaviour of an auction based manufacturing system. While the models discussed here are relatively simple as compared to real life industrial problems, nevertheless they demonstrate the feasibility and the potential behind such an analysis. The prototypes developed here can be made the basis for modelling and analyzing larger and more complex industrial problems in future.

REFERENCES

Ajmone Marsan, A., G. Chiola and A. Fumagalli (1987)." An accurate performance model of CSMA/CD Bus LAN",in *Advances in Petri Nets (1987)*, pp. 146-161.

Al-Jaar,R.Y. and A.A. Desrochers (1990). " Performance evaluation of automated manufacturing systems using generalized stochastic Petri nets", *IEEE Trans. on Robotics and Automation*, 6(6), pp. 621-639.

Alla, H., P. Ladet, J. Martinez, and M. Silva (1985). " Modeling and validation of complex systems by colored Petri nets: application to a flexible manufacturing system", in *Advances in Petri Nets (1984)*, G. Rozenberg, H. Genrich, and G. Roucairol (ed.), Springer-Verlag, pp.15-31.

Balbo, G., G. Chiola, G. Franceschinis, and G.M. Roet (1987)."Generalized stochastic Petri nets for the performance evaluation of FMS", in *Proc. IEEE Int. Conf. on Robotics and Automation*, Raleigh, North Carolina, pp. 1013-1018.

Balbo,G., S.C. Bruell, and S. Ghanta (1988). " Combining queuing networks and generalized stochastic Petri nets for the solution of complex models of system behaviour", *IEEE Trans. on Computers*, 37(10), pp. 1251-1268.

Barson, R.J., N. Huang, M.C. Bonney and M.A. Head (1993)." A Petri Net Method for the Design, Modelling and Simulation of Manufacturing Systems", *Proceedings of the 2nd International Conference on Computer Integrated Manufacturing*, Ed. A. Sen, J. Winsor and R. Gay, pp.417-424.

Bruno, G. and P. Biglia (1985). " Performance evaluation and validation of tool handling in flexible manufacturing systems using Petri nets", *Proc. IEEE 1985 Int. Workshop on Timed Petri Nets*, Torino, Italy, pp. 64-71.

Chin, K.C.K. (1993). " On Petri Net Models of Workstations and Automated Guided Vehicle Systems and their Synthesis for System Level Analysis of Flexible Manufacturing Systems", *Proceedings of the 2nd International Conference on Computer Integrated Manufacturing*, Ed. A. Sen, J. Winsor and R. Gay, pp. 287-295.

Chiola, G. (1985). "A software package for the analysis of generalized stochastic Petri net models", in *Proc. IEEE Int. Workshop on Timed Petri Nets*, Torino, Italy, July, 1985.

Ciardo, G. (1989). *Manual for the SPNP Package*, Duke University, February 1989.

Ciardo, G. and K.S. Trivedi (1991). " A decomposition approach for stochastic Petri net models", in *Proc. of the 4th Int. Workshop on Petri nets and Performance Models*, Melbourne, Australia, pp. 74-83.

Ciardo, G., and J.K. Muppala (1992). *Manual for the SPNP Package, Version 3.1*, Duke University, 1992.

Colom, J.M., M. Silva, and J.L. Villaroel (1986). " On software implementation of Petri nets and colored Petri nets using high-level concurrent language", in *Proc. of the 7th European Workshop on Application and Theory of Petri Nets*, Oxford, pp. 207-241.

Colom,J.M., J.Martinez and M. Silva (1987). " Packages for validating discrete production systems modeled with Petri nets", in *Applied Modeling and Simulation of Technological Systems*, P. Borne and S. G. Tzafestas (eds.), North-Holland, pp. 529-536.

Cruette, D., and J.C. Gentina (1992)." Design and validation of operation sequences in F.M.S", in *Proc. Rensselaer's Third Int. Conf. on Computer Integrated Manufacturing*, Troy, NY, pp. 262-268.

Cumani, A.(1985). " ESP-a package for evaluation of stochastic Petri nets with phase-type distributed transition times", in *Proc. IEEE Int. Workshop on Timed Petri Nets, Torino*, Italy, pp. 144-151.

DiCesare, F. and A.A. Desrochers (1991). " Modeling, control and performance analysis of automated manufacturing systems using Petri nets", in *Control and Dynamic Systems*, C.T. Loendes (ed.), Vol. 47, Academic Press, pp. 121-172.

DiCesare, F. and M.C. Zhou (1992). " Symbolic performance evaluation of concurrent systems combining Petri nets and moment generating functions", To appear in *Control and Dynamic Systems*, C.T. Loendes (ed.), Academic Press.

Dilts,D.M., N.P. Boyd, and H.H. Whorms (1991)." The Evolution of Control Architectures for Automated Control Systems", *Journal of Manufacturing Systems*, 10(1),pp. 63-79.

D'Souza, K.A. and S.K. Khator (1994)." A survey of Petri Net applications in modelling controls for automated manufacturing systems", *Computers in Industry*, Elsevier Science, pp. 5-16.

Dubois, D. and K. Stecke (1983). "Using Petri nets to represent production processes", in *Proc. of the 22nd IEEE Conf. on Decision and Control*, San Antonio, TX,pp. 1062-1067.

Duffie, N.A.,R. Chitturi, and J. Mou (1988)." Fault-Tolerant Heterarchical Control of Heterogeneous manufacturing System Entities", *Journal of Manufacturing Systems*, Vol. 7, No.4, pp. 315-327.

Dugan, J. B., A. Bobbio, A. Ciardo, and K.S. Trivedi (1985). "The design of a unified package for the solution of stochastic Petri Net models", in *Proc. 1985 Int. Workshop on Timed Petri nets*, Torino, Italy, pp. 6-13.

Dunkler, O., C.M. Mitchell, T. Govindaraj, and J.C. Ammons (1988). " The effectiveness of supervisory control strategies in scheduling flexible manufacturing systems", *IEEE trans. on Systems, Man, and Cybernetics*, 18(2), pp. 223-237.

Ezpeleta, J., and J. Martinez (1992). "Formal Specification and Validation in Production Plants" in *Proc. Rensselaer's Third Int. Conf. on Computer Integrated Manufacturing*, Troy, NY, pp. 64-73.

Fukuda, T., S. Takeda and M. Hayashi (1986)." Distributed Expert Systems for Production Control", *Proceedings 1986 IIE Conference*, Dallas, TX, pp. 222-228.

Gentina, J.C. and D. Corbeel (1987). "Colored adaptive structured Petri net: a tool for the automated synthesis of hierarchical control of flexible manufacturing systems", in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Raleigh, NC, pp. 1166-1173.

Gressier, E. (1985). " A stochastic Petri net model for Ethernet", in *Proc. 1985 Int. Workshop on Timed Petri nets*, Torino, Italy, pp. 296-303.

Guo, D.L., F. DiCesare, and M.C. Zhou (1991). " Moment generating function approach to performance analysis of extended stochastic Petri nets", in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, pp. 1309-1314.

Hillion ,H.P. and J.M. Proth (1989)." Performance evaluation of job shop systems using timed-event-graphs",*IEEE Trans. on Automatic Control*, 34(1), pp. 3-9.

Holliday, M.A. and M.K. Vernon (1985). " A generalized timed Petri net model for performance analysis", in *Proc. 1985 IEEE Int. Workshop on Timed Petri Nets*, Torino, Italy, pp. 181-190.

Inamasu, R.Y. and A.J.V. Porto (1993). "A production planning system based on a Temporal Petri Net", *Proceedings of the 2nd International Conference on Computer Integrated Manufacturing*, Ed. A. Sen, J. Winsor and R. Gay, pp. 551-557.

Jungnitz, H.J. and A.A. Desrochers (1991)." Flow equivalent nets for the performance analysis of generalized stochastic Petri nets", in *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, pp. 122-127.

Jungnitz, H.J. (1992). Approximation Methods for Stochastic Petri Nets, Doctoral Dissertation ,ECSE, Rensselaer Polytechnic Institute, Troy, NY.

Jungnitz, H.J. and A.A. Desrochers (1991). "Flow equivalent nets for the performance analysis of generalized stochastic Petri nets", in *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, pp. 122-127.

Krogh, B.H. and R.S. Sreenivas (1987). "Essentially decision free Petri nets for real-time resource allocation", in *Proc. of IEEE Int. Conf. Robotics and Automation*, Raleigh, NC, pp. 1005-1011.

Lewis, W., M.M. Barash and J.J. Solberg (1982)." Single Queue Management of a Job Shop as Implemented by a Data Flow Architecture", *Proceedings of the 23rd International Machine Tool Design and Research Conference*, Manchester, pp. 415-420, September 1982.

Ma, J. and M.C. Zhou (1992). "Performance evaluation of discrete event systems via stepwise reduction and approximation of stochastic Petri nets", To appear in *Proc. 31st IEEE Int. Conf. on Decision and Control*, Tucson, AZ.

Martinez, J., H. Alla, and M. Silva (1986). " Petri nets for the specifications of FMSs", in *Modeling and Design of Flexible Manufacturing Systems*, A. Kusiak (ed.), Elsevier Science Publishers, Amsterdam, pp. 389-406.

Martinez, J., P. Muro and M. Silva (1987)." Modeling, validation and software implementation of production systems using high level Petri nets", in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Raleigh, North Carolina, pp. 1180-1185.

Narahari, Y. and N. Viswanadham (1985). " A Petri net approach to the modeling and analysis of flexible manufacturing systems", *Annals of Operation Research*, 3, pp. 449-472.

Parunak, H. (1987)." Manufacturing Experience with the Contract Net ", *Distributed Artificial Intelligence*, Michael M. Huhns (ed.), Pitman, London, pp. 285-310, 1987.

Parunak, H., B.W. vanDyke, J.K. Irish, and P.W. Lozo (1985)." Fractal Actors for Distributed Manufacturing Control", *Proceedings of the IEEE Second Conference on AI Applications*, Miami Beach, FL, pp. 653-660, December 1985.

Pimental, J.R. (1990). Communication Networks for Manufacturing, Prentice hall, Englewood Cliffs.

Ramaswamy, S. and K.P. Valanavis (1993). " Modelling, Simulation and Analysis of Failures in a Materials Handling System with Extended Petri Nets", pp. 323-330.

Raju, K.R. and O.V.K. Chetty (1993). " Priority Nets for Scheduling Flexible Manufacturing Systems", *Journal of Manufacturing Systems*, Vol 12, No.4, pp. 326-340.

Seidmann, A. and S.Y. Nof (1989). " Operational analysis of an autonomous assembly robotic station", *IEEE Trans. on Robotics and Automation*, 5(1), pp. 4-15.

Shaw, M.J., and A.B. Whinston (1985)." Automatic planning and Flexible Scheduling: A Knowledge base", *IEEE International Conference on Robotics and Automation*, St. Louis, Missouri, pp. 890-894, 1985.

Smith, R.G. (1980)." The contract net protocol:high-level communication and control in a distributed problem solver", *IEEE Trans. Computers*, C-29: 1104-1113.

Upton, D.M., M.M. Barash and A.M. Matheson (1991)." Architectures and Auctions in Manufacturing", *International Journal of Computer Integrated Manufacturing*, 1991, Vol.4. No.1, 23-33.

Upton, D.M. (1992)." A Flexible Structure for Computer-Controlled manufacturing Systems", *Manufacturing review*, Vol.5, No.1, pp.58-72, March 1992.

Van der Aalst, W.M.P. (1994)." Putting High Level Petri Nets to work in the industry", *Computers in Industry*, pp. 45-54.

Valette, R., M. Courvoisier, and D. Mayeux (1982)."Control of flexible production systems and Petri nets", in *Informatik Fachberichte 66*, Springer Verlag, pp.264-267.

Valette, R. (1987)." Nets in production systems", in *Advances in Petri Nets 1986*, W.Brauer, W. Reisig, and G. Rozenberg (eds.), Vol. 255, Part I, Springer- Verlag, pp. 191-217.

Viswanadham, N. and Y. Narahari (1987). " Colored Petri net models for the automated manufacturing systems", in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Raleigh, North Carolina, pp. 1985-1990.

Viswanadham, N. and Y. Narahari (1988). "Stochastic Petri net models for the performance evaluation of automated manufacturing systems", *Information and Decision technologies*, 14, North-Holland, pp. 125-142.

Viswanadham, N. and T.L. Johnson (1990)."Performance analysis of automated manufacturing systems with blocking and deadlock", in *Rensselaer's 2nd Int. Conf. on Computer Integrated Manufacturing*, Troy, NY, pp. 64-68.

Viswanadham, N. and Y. Narahari (1992). Performance Modelling of Automated Manufacturing Systems, Prentice Hall, NJ.

Wang, F.Y. and K. Gildea (1989). " A colored Petri net model for connection management services in manufacturing message specification (MMS)", *Computer Communication Review*, 19(3), pp. 76-98.

Wang, L.C. (1993)." The Development of an Object Oriented Petri Net Cell Control Model", *Proceedings of the 2nd International Conference on Computer Integrated Manufacturing*, Ed. A. Sen, J. Winsor and R. Gay, pp. 398-405.

Watson III, J.F. and A.A. Desrochers (1991). " Applying GSPN's to manufacturing systems containing non-exponential transition functions", in *Proc. of IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, pp. 366-371.

Yang, E.H., M.M. Barash and D.M. Upton (1993)."Accommodation of priority parts in a distributed computer controlled manufacturing system with aggregate bidding schemes", in *Proc. of 2nd Industrial Engineering Research Conference*, pp. 827-831.

Yao, D.D. and J.A. Buzacott (1985). "Modeling the performance of flexible manufacturing systems", *J. Prod. Res.*, 23(5), pp. 945-959.

Zhou, M.C. and F. DiCesare (1989). "Adaptive design of Petri net controllers for error recovery in automated manufacturing systems", in *IEEE Trans on Systems, Man and Cybernetics*, 19(5), pp. 963-973.

Zhou, M.C., F. DiCesare, and D. Guo (1990). "Modeling and performance analysis of a resource-sharing manufacturing system using Stochastic Petri nets", in *Proc. IEEE Int. Symp. on Intelligent Control*, Philadelphia, PA., pp. 1005-1010.

Zhou, M.C. and M.C. Leu (1991). " Modeling and performance analysis of a flexible PCB assembly station using Petri nets", *Trans. of the ASME, Journal of Electronic Packaging*, 113(4), pp. 410-416.

Zhou, M.C., D.L. Guo and F. DiCesare (1992)." Integration of Petri nets and moment generating function approaches for the system performance evaluation", *J. of Systems Integration.*

Zhou, Q. (1993). " Performance Evaluation of and Auction Based Manufacturing Cell using Generalized Stochastic Petri Nets", Masters Thesis, Department of Industrial and Manufacturing Systems Engineering, University of Windsor.

Zhou, Q., S.P. Dutta and M.H. Wang (1994)." Performance Evaluation of an Auction Based FMC using GSPN", *International Journal of Manufacturing Systems Design*, under review.

APPENDIX 1: CODE FOR STEADY STATE ANALYSIS OF BASIC MODEL

```
# include "user.h"
# include <math.h>
# define min(x,y) ((x<y)?x:y)

int sh,m,n,bf,s,q;
float  rateup1[3],ratedp1[3],rateup2[3],ratedp2[3],rateup3[3],ratedp3[3],val1,val2,lamda;
int count0,count1,count2,count3;
parameters()

{
 FILE *fp;
  int i,j;

        iopt(IOP_METHOD,VAL_GASEI);
        iopt(IOP_PR_FULL_MARK,VAL_YES);
        iopt(IOP_PR_MARK_ORDER,VAL_CANONIC);
        iopt(IOP_PR_MC_ORDER,VAL_TOFROM);
        iopt(IOP_PR_MC,VAL_NO);
        iopt(IOP_PR_PROB,VAL_NO);
        iopt(IOP_MC,VAL_CTMC);
        iopt(IOP_PR_RSET,VAL_YES);
        iopt(IOP_PR_RGRAPH,VAL_YES);
        iopt(IOP_ITERATIONS,2000);
        fopt(FOP_PRECISION,1.0e-3);
        fopt(FOP_ABS_RET_M0,0.0);
 }

 /* ******* STEADY STATE ANALYSIS ************  */
 /*          SQL rule              */

 enabling_type buffr_choose(i)
 {
 float PT1,PT;
 int k,h,K;
```

```
if(mark_1("pc",0)==1 || mark_1("pc",0)==0){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc",0)==2){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc",0)==3){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
```

```
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}


enabling_type buffr_choose2(i)
{

float PT1,PT;
int k,h,K;

if(mark_1("pc2",0)==1 || mark_1("pc2",0)==0){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc2",0)==2){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
```

```
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc2",0)==3){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}

enabling_type buffr_choose3(i)
{

float PT1,PT;
int k,h,K;

if(mark_1("pc3",0)==1 || mark_1("pc3",0)==0){
/* choose from 4,5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
```

```
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc3",0)==2){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}


if(mark_1("pc3",0)==3){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}
```

```
enabling_type upstream_choose(i)
{

float PT1,PT;
int k,b K;

if(mark_1("pc",0)==0 || mark_1("pc",0)==3){
/* choose from 2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc",0)==1){
/* choose from 1,2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=2;k=k+2){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc",0)==2){
/* choose from 2 & 3*/
PT1=1.0e+30;PT=0.0;
```

```
for(k=1;k<=2;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}

enabling_type upstream_choose2(i)
{

float PT1,PT;
int k,h,K;

if(mark_1("pc2",0)==0 || mark_1("pc2",0)==3){
/* choose from 2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc2",0)==1){
/* choose from 1 & 2*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=1;k++){
```

```
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}


if(mark_1("pc2",0)==2){
/* choose from 1 & 2*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=1;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}

enabling_type upstream_choose3(i)
{

float PT1,PT,PTP;
int k,h,K;

float a1,a2,a3;
if(mark_1("pc3",0)==0 || mark_1("pc3",0)==3){
/* choose from  3*/
PT1=1.0e+30;PT=0.0;
```

```
for(k=2;k<=2;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc3",0)==1){
/* choose from 2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc3",0)==2){
/* choose from 1 & 2*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=1;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
```

```
else return (0);
}
else return (0);
}

arc_car()

{
  if(mark_1("pc",0)>=3) return (3);
  else return (0);
}

arc_car2()

{
  if(mark_1("pc2",0)>=3) return (3);
  else return (0);
}

arc_car3()

{
  if(mark_1("pc3",0)>=3) return (3);
  else return (0);
}


net()
{
 int i,j;
 char auxplace[20],auxtrans[20];
 FILE *fp;

/*places and transitions*/

 place_1("pc",2);
 place_1("psf",3);
 place_1("pu",3);
 place_1("pd",3);
```

```
trans_2("tui",3,2);
trans_1("tbufs",3);
trans_1("tdi",4);
place_1("pcomm",3);
trans_1("tcomm",2);

place_1("pc2",2);
place_1("psf2",3);
place_1("pu2",3);
place_1("pd2",3);
trans_2("tui2",3,2);
trans_1("tbufs2",3);
trans_1("tdi2",4);
place_1("pcomm2",3);
trans_1("tcomm2",2);

place_1("pc3",2);
place_1("psf3",3);
place_1("pu3",3);
place_1("pd3",3);
trans_2("tui3",3,2);
trans_1("tbufs3",3);
trans_1("tdi3",4);
place_1("pcomm3",3);
trans_1("tcomm3",2);

/* Defining transition rates */

rateup1[0]=0.025; rateup1[1]=0.05;  rateup1[2]=0.025;
ratedp1[0]=0.05;  ratedp1[1]=0.025; ratedp1[2]=0.025;

probval_2("tui",0,0,1.0);
probval_2("tui",1,0,1.0);
probval_2("tui",2,0,1.0);
rateval_2("tui",0,1,0.025);
rateval_2("tui",1,1,0.05);
rateval_2("tui",2,1,0.025);
probval_1("tbufs",ALL,1.0);
rateval_1("tdi",0,0.05);
```

```
rateval_1("tdi",1,0.025);
rateval_1("tdi",2,0.025); rateval_1("tdi",3,1.0);
rateval_1("tcomm",ALL,2.0);

rateup2[0]=0.02;  rateup2[1]=0.02;  rateup2[2]=0.03;
ratedp2[0]=0.200;  ratedp2[1]=0.40;  ratedp2[2]=0.16;


probval_2("tui2",0,0,1.0);
probval_2("tui2",1,0,1.0);
probval_2("tui2",2,0,1.0);
rateval_2("tui2",0,1,0.02);
rateval_2("tui2",1,1,0.02);
rateval_2("tui2",2,1,0.03);
probval_1("tbufs2",ALL,1.0);
rateval_1("tdi2",0,0.2);
rateval_1("tdi2",1,0.4);
rateval_1("tdi2",2,0.16);
rateval_1("tdi2",3,1.0);
rateval_1("tcomm2",ALL,2.0);

rateup3[0]=0.9;  rateup3[1]=0.9;  rateup3[2]=0.9;
ratedp3[0]=0.1;  ratedp3[1]=0.04;  ratedp3[2]=0.05;


probval_2("tui3",0,0,1.0);
probval_2("tui3",1,0,1.0);
probval_2("tui3",2,0,1.0);
rateval_2("tui3",0,1,0.9);
rateval_2("tui3",1,1,0.9);
rateval_2("tui3",2,1,0.9);
probval_1("tbufs3",ALL,1.0);
rateval_1("tdi3",0,0.1);
rateval_1("tdi3",1,0.04);
rateval_1("tdi3",2,0.05);
rateval_1("tdi3",3,1.0);
rateval_1("tcomm3",ALL,2.0);
```

```
/* SQL rule */

for(i=0;i<=2;i++){
rateup1[i]=1.0;
rateup2[i]=1.0;
rateup3[i]=1.0;
ratedp1[i]=1.0;
ratedp2[i]=1.0;
ratedp3[i]=1.0;
}


priority_2("tui",ALL,0,2);
priority_1("tbufs",ALL,1);

priority_2("tui2",ALL,0,2);
priority_1("tbufs2",ALL,1);

priority_2("tui3",ALL,0,2);
priority_1("tbufs3",ALL,1);

priority_1("tdi",3,4);
priority_1("tdi2",3,5);
priority_1("tdi3",3,6);

/*set initial markings */

init_1( "psf",2,3);

init_1("pcomm",0,1);
init_1("pcomm2",0,1);
init_1("pcomm3",0,1);

/* psf to upstream */

iarc_2_1("tui",0,0,"psf",0);
iarc_2_1("tui",1,0,"psf",0);
iarc_2_1("tui",2,0,"psf",0);
```

```
iarc_2_1("tui2",0,0,"psf2",0);
iarc_2_1("tui2",1,0,"psf2",0);
iarc_2_1("tui2",2,0,"psf2",0);

iarc_2_1("tui3",0,0,"psf3",0);
iarc_2_1("tui3",1,0,"psf3",0);
iarc_2_1("tui3",2,0,"psf3",0);


iarc_1_1("tdi",3,"psf",2);
oarc_1_1("tdi",3,"psf",0);

iarc_1_1("tdi2",3,"psf",2);
oarc_1_1("tdi2",3,"psf2",0);

iarc_1_1("tdi3",3,"psf",2);
oarc_1_1("tdi3",3,"psf3",0);

/* upstream machine */

oarc_2_1("tui",0,0,"pu",0);
oarc_2_1("tui",1,0,"pu",1);
oarc_2_1("tui",2,0,"pu",2);
iarc_2_1("tui",0,1,"pu",0);
iarc_2_1("tui",1,1,"pu",1);
iarc_2_1("tui",2,1,"pu",2);

oarc_2_1("tui",0,1,"psf",1);
oarc_2_1("tui",1,1,"psf",1);
oarc_2_1("tui",2,1,"psf",1);


oarc_2_1("tui2",0,0,"pu2",0);
oarc_2_1("tui2",1,0,"pu2",1);
oarc_2_1("tui2",2,0,"pu2",2);
iarc_2_1("tui2",0,1,"pu2",0);
iarc_2_1("tui2",1,1,"pu2",1);
iarc_2_1("tui2",2,1,"pu2",2);
```

```
oarc_2_1("tui2",0,1,"psf2",1);
oarc_2_1("tui2",1,1,"psf2",1);
oarc_2_1("tui2",2,1,"psf2",1);


oarc_2_1("tui3",0,0,"pu3",0);
oarc_2_1("tui3",1,0,"pu3",1);
oarc_2_1("tui3",2,0,"pu3",2);
iarc_2_1("tui3",0,1,"pu3",0);
iarc_2_1("tui3",1,1,"pu3",1);
iarc_2_1("tui3",2,1,"pu3",2);

oarc_2_1("tui3",0,1,"psf3",1);
oarc_2_1("tui3",1,1,"psf3",1);
oarc_2_1("tui3",2,1,"psf3",1);


/* counter */

oarc_2_1("tui",0,0, "pc",0);
oarc_2_1("tui",1,0, "pc",0);
oarc_2_1("tui",2,0, "pc",0);

viarc_1_1("tcomm",0,"pc",0,arc_car);

/* INHIBITOR ARCS */

mharc_1_1("tdi2",3,"psf2",0,1);
mharc_1_1("tdi3",3,"psf3",0,1);

oarc_2_1("tui2",0,0, "pc2",0);
oarc_2_1("tui2",1,0, "pc2",0);
oarc_2_1("tui2",2,0, "pc2",0);

viarc_1_1("tcomm2",0,"pc2",0,arc_car);


oarc_2_1("tui3",0,0, "pc3",0);
oarc_2_1("tui3",1,0, "pc3",0);
```

```
oarc_2_1("tui3",2,0, "pc3",0);

viarc_1_1("tcomm3",0,"pc3",0,arc_car3);


/* upstream machine to communication subnet */

iarc_1_1("tcomm",0, "psf",1);
iarc_1_1("tcomm",0, "pcomm",0);
oarc_1_1("tcomm",0, "pcomm",1);
iarc_1_1("tcomm",1, "pcomm",1);
oarc_1_1("tcomm",1, "pcomm",0);
oarc_1_1("tcomm",1, "pcomm",2);

iarc_1_1("tcomm2",0, "psf2",1);
iarc_1_1("tcomm2",0, "pcomm2",0);
oarc_1_1("tcomm2",0, "pcomm2",1);
iarc_1_1("tcomm2",1, "pcomm2",1);
oarc_1_1("tcomm2",1, "pcomm2",0);
oarc_1_1("tcomm2",1, "pcomm2",2);

iarc_1_1("tcomm3",0, "psf3",1);
iarc_1_1("tcomm3",0, "pcomm3",0);
oarc_1_1("tcomm3",0, "pcomm3",1);
iarc_1_1("tcomm3",1, "pcomm3",1);
oarc_1_1("tcomm3",1, "pcomm3",0);
oarc_1_1("tcomm3",1, "pcomm3",2);


/* communication subnet , downstream m/c */

iarc_1_1("tbufs",0,"pcomm",2);
iarc_1_1("tbufs",1,"pcomm",2);
iarc_1_1("tbufs",2,"pcomm",2);

iarc_1_1("tbufs2",0,"pcomm2",2);
iarc_1_1("tbufs2",1,"pcomm2",2);
iarc_1_1("tbufs2",2,"pcomm2",2);
```

```
iarc_1_1("tbufs3",0,"pcomm3",2);
iarc_1_1("tbufs3",1,"pcomm3",2);
iarc_1_1("tbufs3",2,"pcomm3",2);


/* downstream m/c */

oarc_1_1("tbufs",0,"pd",0);
oarc_1_1("tbufs",1,"pd",1);
oarc_1_1("tbufs",2,"pd",2);
iarc_1_1("tdi",0,"pd",0);
iarc_1_1("tdi",1,"pd",1);
iarc_1_1("tdi",2,"pd",2);
oarc_1_1("tdi",0,"psf",0);
oarc_1_1("tdi",1,"psf",0);
oarc_1_1("tdi",2,"psf",0);

oarc_1_1("tbufs2",0,"pd2",0);
oarc_1_1("tbufs2",1,"pd2",1);
oarc_1_1("tbufs2",2,"pd2",2);
iarc_1_1("tdi2",0,"pd2",0);
iarc_1_1("tdi2",1,"pd2",1);
iarc_1_1("tdi2",2,"pd2",2);
oarc_1_1("tdi2",0,"psf2",0);
oarc_1_1("tdi2",1,"psf2",0);
oarc_1_1("tdi2",2,"psf2",0);

oarc_1_1("tbufs3",0,"pd3",0);
oarc_1_1("tbufs3",1,"pd3",1);
oarc_1_1("tbufs3",2,"pd3",2);
iarc_1_1("tdi3",0,"pd3",0);
iarc_1_1("tdi3",1,"pd3",1);
iarc_1_1("tdi3",2,"pd3",2);
oarc_1_1("tdi3",0,"psf3",0);
oarc_1_1("tdi3",1,"psf3",0);
oarc_1_1("tdi3",2,"psf3",0);
```

```
/* marking dependent enabling functions */

for (i=0;i<=2;i++){
enabling_1("tbufs",i,buffr_choose);
}

for (i=0;i<=2;i++){
enabling_1("tbufs2",i,buffr_choose2);
}

for (i=0;i<=2;i++){
enabling_1("tbufs3",i,buffr_choose3);
}

for (i=0;i<=2;i++){
enabling_2("tui",i,0,upstream_choose);
}

for (i=0;i<=2;i++){
enabling_2("tui2",i,0,upstream_choose2);
}

for (i=0;i<=2;i++){
enabling_2("tui3",i,0,upstream_choose3);
}

}

reward_type umc() { return(mark_1("pcomm",1));}
reward_type dmc() { return(mark_1("pcomm",2));}
reward_type pmc() { return(mark_1("psf",2));}
reward_type umc2() { return(mark_1("pcomm2",1));}
reward_type dmc2() { return(mark_1("pcomm2",2));}
reward_type umc3() { return(mark_1("pcomm3",1));}
reward_type dmc3() { return(mark_1("pcomm3",2));}

reward_type plu1() { return(mark_1("pu",0)*5);}
reward_type plu2() { return(mark_1("pu",1)*5);}
reward_type plu3() { return(mark_1("pu",2)*5);}
```

```
reward_type p1d1() { return(mark_1("pd",0)*5);}
reward_type p1d2() { return(mark_1("pd",1)*5);}
reward_type p1d3() { return(mark_1("pd",2)*5);}

reward_type p2u1() { return(mark_1("pu2",0)*26);}
reward_type p2u2() { return(mark_1("pu2",1)*26);}
reward_type p2u3() { return(mark_1("pu2",2)*26);}
reward_type p2d1() { return(mark_1("pd2",0)*26);}
reward_type p2d2() { return(mark_1("pd2",1)*26);}
reward_type p2d3() { return(mark_1("pd2",2)*26);}

reward_type p3u1() { return(mark_1("pu3",0)*13);}
reward_type p3u2() { return(mark_1("pu3",1)*13);}
reward_type p3u3() { return(mark_1("pu3",2)*13);}
reward_type p3d1() { return(mark_1("pd3",0)*13);}
reward_type p3d2() { return(mark_1("pd3",1)*13);}
reward_type p3d3() { return(mark_1("pd3",2)*13);}

ac_final() {
int const;
pr_mc_info();
pr_std_average();
const=10000;

}

assert() {
return (RES_NOERR);
}

ac_init() {
fprintf(stdout,"\n Petri Net Model of an Auction Based Manufacturing System \n\n");
pr_net_info();
}

ac_reach() {
fprintf(stdout,"\n The reachability graph has been generated \n\n");
pr_rg_info();
}
```

APPENDIX 2: CODE FOR TRANSIENT ANALYSIS OF BASIC MODEL

```
# include "user.h"
# include <math.h>
# define min(x,y) ((x<y)?x:y)

int sh,m,n,bf,s,q;
float  rateup1[3],ratedp1[3],rateup2[3],ratedp2[3],rateup3[3],ratedp3[3],val1,val2,lamda;
int count0,count1,count2,count3;
parameters()

{
 FILE *fp;
  int i,j;

        iopt(IOP_METHOD,VAL_TSUNIF);
        iopt(IOP_PR_FULL_MARK,VAL_YES);
        iopt(IOP_PR_MARK_ORDER,VAL_CANONIC);
        iopt(IOP_PR_MC_ORDER,VAL_TOFROM);
        iopt(IOP_PR_MC,VAL_NO);
        iopt(IOP_PR_PROB,VAL_NO);
        iopt(IOP_MC,VAL_CTMC);
        iopt(IOP_PR_RSET,VAL_YES);
        iopt(IOP_PR_RGRAPH,VAL_YES);
         iopt(IOP_ITERATIONS,20000);
         iopt(IOP_SENSITIVITY,VAL_YES);
         fopt(FOP_PRECISION,1.0e-3);
        fopt(FOP_ABS_RET_M0,0.0);

}

/* ******* TRANSIENT ANALYSIS ***************  */
/*          SQL RULE                */


enabling_type buffr_choose(i)
{
float PT1,PT;
int k,h,K;
```

```
if(mark_1("pc",0)==1){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc",0)==2){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc",0)==3){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
```

```
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}


enabling_type buffr_choose2(i)
{

float PT1,PT;
int k,h,K;

if(mark_1("pc2",0)==1){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc2",0)==2){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
```

```
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc2",0)==3){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}

enabling_type buffr_choose3(i)
{

float PT1,PT;
int k,h,K;

if(mark_1("pc3",0)==1){
/* choose from 4,5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
```

```
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc3",0)==2){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}


if(mark_1("pc3",0)==3){
/* choose from 5 & 6*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=mark_1("pd",k)*(1/ratedp1[k])*5+mark_1("pd2",k)*(1/ratedp2[k])*26+mark_1(
"pd3",k)*(1/ratedp3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}

enabling_type upstream_choose(i)
```

```
{

float PT1,PT;
int k,h,K;

if(mark_1("pc",0)==0){
/* choose from 2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc",0)==1){
/* choose from 1,2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=2;k=k+2){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc",0)==2){
/* choose from 2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
```

```
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}

enabling_type upstream_choose2(i)
{

float PT1,PT;
int k,h,K;

if(mark_1("pc2",0)==0){
/* choose from 2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc2",0)==1){
/* choose from 1 & 2*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=1;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
```

```
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}


if(mark_1("pc2",0)==2){
/* choose from 1 & 2*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=1;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}
else return (0);
}

enabling_type upstream_choose3(i)
{

float PT1,PT,PTP;
int k,h,K;

float a1,a2,a3;
if(mark_1("pc3",0)==0){
/* choose from 3*/
PT1=1.0e+30;PT=0.0;
for(k=2;k<=2;k++){
```

```
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc3",0)==1){
/* choose from 2 & 3*/
PT1=1.0e+30;PT=0.0;
for(k=1;k<=2;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
}

if(mark_1("pc3",0)==2){
/* choose from 1 & 2*/
PT1=1.0e+30;PT=0.0;
for(k=0;k<=1;k++){
PT=(mark_1("pu",k)/rateup1[k])*5+(mark_1("pu2",k)/rateup2[k])*26+(mark_1("pu
3",k)/rateup3[k])*13;
if(PT<=PT1){
PT1=PT;
K=k;
}
}
if(K==i) return (1);
else return (0);
```

```
}
else return (0);
}

arc_car()

{
  if(mark_1("pc",0)>=2) return (2);
  else return (0);
}

arc_car2()

{
  if(mark_1("pc2",0)>=2) return (2);
  else return (0);
}


arc_car3()

{
  if(mark_1("pc3",0)>=2) return (2);
  else return (0);
}


net()
{
 int i,j;
 char auxplace[20],auxtrans[20];
 FILE *fp;

 /*places and transitions*/

 place_1("pc",2);
 place_1("psf",3);
 place_1("pu",3);
 place_1("pd",3);
```

```
trans_2("tui",3,2);
trans_1("tbufs",3);
trans_1("tdi",4);
place_1("pcomm",3);
trans_1("tcomm",2);

place_1("pc2",2);
place_1("psf2",3);
place_1("pu2",3);
place_1("pd2",3);
trans_2("tui2",3,2);
trans_1("tbufs2",3);
trans_1("tdi2",4);
place_1("pcomm2",3);
trans_1("tcomm2",2);

place_1("pc3",2);
place_1("psf3",3);
place_1("pu3",3);
place_1("pd3",3);
trans_2("tui3",3,2);
trans_1("tbufs3",3);
trans_1("tdi3",4);
place_1("pcomm3",3);
trans_1("tcomm3",2);

/* Defining transition rates */

rateup1[0]=0.025; rateup1[1]=0.05;  rateup1[2]=0.025;
ratedp1[0]=0.05;  ratedp1[1]=0.025; ratedp1[2]=0.025;

probval_2("tui",0,0,1.0);
probval_2("tui",1,0,1.0);
probval_2("tui",2,0,1.0);
rateval_2("tui",0,1,0.025);
rateval_2("tui",1,1,0.05);
rateval_2("tui",2,1,0.025);
probval_1("tbufs",ALL,1.0);
rateval_1("tdi",0,0.05);
```

```
rateval_1("tdi",1,0.025);
rateval_1("tdi",2,0.025);
rateval_1("tdi",3,1.0);
rateval_1("tcomm",ALL,2.0);

rateup2[0]=0.02;   rateup2[1]=0.02;  rateup2[2]=0.03;
ratedp2[0]=0.200;  ratedp2[1]=0.40;  ratedp2[2]=0.16;


probval_2("tui2",0,0,1.0);
probval_2("tui2",1,0,1.0);
probval_2("tui2",2,0,1.0);
rateval_2("tui2",0,1,0.02);
rateval_2("tui2",1,1,0.02);
rateval_2("tui2",2,1,0.03);
probval_1("tbufs2",ALL,1.0);
rateval_1("tdi2",0,0.2);
rateval_1("tdi2",1,0.4);
rateval_1("tdi2",2,0.16);
rateval_1("tdi2",3,1.0);
rateval_1("tcomm2",ALL,2.0);

rateup3[0]=0.9;  rateup3[1]=0.9;  rateup3[2]=0.9;
ratedp3[0]=0.1;  ratedp3[1]=0.04; ratedp3[2]=0.05;

probval_2("tui3",0,0,1.0);
probval_2("tui3",1,0,1.0);
probval_2("tui3",2,0,1.0);
rateval_2("tui3",0,1,0.9);
rateval_2("tui3",1,1,0.9);
rateval_2("tui3",2,1,0.9);
probval_1("tbufs3",ALL,1.0);
rateval_1("tdi3",0,0.1);
rateval_1("tdi3",1,0.04);
rateval_1("tdi3",2,0.05); /* 0.05  0.025 */
rateval_1("tdi3",3,1.0);
rateval_1("tcomm3",ALL,2.0);
```

```
/* SQL rule */

for(i=0;i<=2;i++){
rateup1[i]=1.0;
rateup2[i]=1.0;
rateup3[i]=1.0;
ratedp1[i]=1.0;
ratedp2[i]=1.0;
ratedp3[i]=1.0;
}


priority_2("tui",ALL,0,2);
priority_1("tbufs",ALL,1);

priority_2("tui2",ALL,0,2);
priority_1("tbufs2",ALL,1);

priority_2("tui3",ALL,0,2);
priority_1("tbufs3",ALL,1);

priority_1("tdi",3,4);
priority_1("tdi2",3,5);
priority_1("tdi3",3,6);

/*set initial markings */

init_1( "psf",2,3);

init_1("pcomm",0,1);
init_1("pcomm2",0,1);
init_1("pcomm3",0,1);

/* psf to upstream */

iarc_2_1("tui",0,0,"psf",0);
iarc_2_1("tui",1,0,"psf",0);
iarc_2_1("tui",2,0,"psf",0);
```

```
iarc_2_1("tui2",0,0,"psf2",0);
iarc_2_1("tui2",1,0,"psf2",0);
iarc_2_1("tui2",2,0,"psf2",0);

iarc_2_1("tui3",0,0,"psf3",0);
iarc_2_1("tui3",1,0,"psf3",0);
iarc_2_1("tui3",2,0,"psf3",0);


iarc_1_1("tdi",3,"psf",2);
oarc_1_1("tdi",3,"psf",0);

iarc_1_1("tdi2",3,"psf",2);
oarc_1_1("tdi2",3,"psf2",0);

iarc_1_1("tdi3",3,"psf",2);
oarc_1_1("tdi3",3,"psf3",0);

/* upstream machine */

oarc_2_1("tui",0,0,"pu",0);
oarc_2_1("tui",1,0,"pu",1);
oarc_2_1("tui",2,0,"pu",2);
iarc_2_1("tui",0,1,"pu",0);
iarc_2_1("tui",1,1,"pu",1);
iarc_2_1("tui",2,1,"pu",2);

oarc_2_1("tui",0,1,"psf",1);
oarc_2_1("tui",1,1,"psf",1);
oarc_2_1("tui",2,1,"psf",1);


oarc_2_1("tui2",0,0,"pu2",0);
oarc_2_1("tui2",1,0,"pu2",1);
oarc_2_1("tui2",2,0,"pu2",2);
iarc_2_1("tui2",0,1,"pu2",0);
iarc_2_1("tui2",1,1,"pu2",1);
iarc_2_1("tui2",2,1,"pu2",2);
```

```
oarc_2_1("tui2",0,1,"psf2",1);
oarc_2_1("tui2",1,1,"psf2",1);
oarc_2_1("tui2",2,1,"psf2",1);


oarc_2_1("tui3",0,0,"pu3",0);
oarc_2_1("tui3",1,0,"pu3",1);
oarc_2_1("tui3",2,0,"pu3",2);
iarc_2_1("tui3",0,1,"pu3",0);
iarc_2_1("tui3",1,1,"pu3",1);
iarc_2_1("tui3",2,1,"pu3",2);

oarc_2_1("tui3",0,1,"psf3",1);
oarc_2_1("tui3",1,1,"psf3",1);
oarc_2_1("tui3",2,1,"psf3",1);


/* counter */

oarc_2_1("tui",0,0, "pc",0);
oarc_2_1("tui",1,0, "pc",0);
oarc_2_1("tui",2,0, "pc",0);

mharc_1_1("tdi2",3,"psf2",0,1);
mharc_1_1("tdi3",3,"psf3",0,1);

/* ADDITIONAL INHIBITOR ARCS ABOVE */

oarc_2_1("tui2",0,0, "pc2",0);
oarc_2_1("tui2",1,0, "pc2",0);
oarc_2_1("tui2",2,0, "pc2",0);


oarc_2_1("tui3",0,0, "pc3",0);
oarc_2_1("tui3",1,0, "pc3",0);
oarc_2_1("tui3",2,0, "pc3",0);
```

```
/* upstream machine to communication subnet */

iarc_1_1("tcomm",0, "psf",1);
iarc_1_1("tcomm",0, "pcomm",0);
oarc_1_1("tcomm",0, "pcomm",1);
iarc_1_1("tcomm",1, "pcomm",1);
oarc_1_1("tcomm",1, "pcomm",0);
oarc_1_1("tcomm",1, "pcomm",2);

iarc_1_1("tcomm2",0, "psf2",1);
iarc_1_1("tcomm2",0, "pcomm2",0);
oarc_1_1("tcomm2",0, "pcomm2",1);
iarc_1_1("tcomm2",1, "pcomm2",1);
oarc_1_1("tcomm2",1, "pcomm2",0);
oarc_1_1("tcomm2",1, "pcomm2",2);

iarc_1_1("tcomm3",0, "psf3",1);
iarc_1_1("tcomm3",0, "pcomm3",0);
oarc_1_1("tcomm3",0, "pcomm3",1);
iarc_1_1("tcomm3",1, "pcomm3",1);
oarc_1_1("tcomm3",1, "pcomm3",0);
oarc_1_1("tcomm3",1, "pcomm3",2);

/* communication subnet , downstream m/c */

iarc_1_1("tbufs",0,"pcomm",2);
iarc_1_1("tbufs",1,"pcomm",2);
iarc_1_1("tbufs",2,"pcomm",2);

iarc_1_1("tbufs2",0,"pcomm2",2);
iarc_1_1("tbufs2",1,"pcomm2",2);
iarc_1_1("tbufs2",2,"pcomm2",2);

iarc_1_1("tbufs3",0,"pcomm3",2);
iarc_1_1("tbufs3",1,"pcomm3",2);
iarc_1_1("tbufs3",2,"pcomm3",2);
```

```
/* downstream m/c */

oarc_1_1("tbufs",0,"pd",0);
oarc_1_1("tbufs",1,"pd",1);
oarc_1_1("tbufs",2,"pd",2);
iarc_1_1("tdi",0,"pd",0);
iarc_1_1("tdi",1,"pd",1);
iarc_1_1("tdi",2,"pd",2);
oarc_1_1("tdi",0,"psf",0);
oarc_1_1("tdi",1,"psf",0);
oarc_1_1("tdi",2,"psf",0);

oarc_1_1("tbufs2",0,"pd2",0);
oarc_1_1("tbufs2",1,"pd2",1);
oarc_1_1("tbufs2",2,"pd2",2);
iarc_1_1("tdi2",0,"pd2",0);
iarc_1_1("tdi2",1,"pd2",1);
iarc_1_1("tdi2",2,"pd2",2);
oarc_1_1("tdi2",0,"psf2",0);
oarc_1_1("tdi2",1,"psf2",0);
oarc_1_1("tdi2",2,"psf2",0);

oarc_1_1("tbufs3",0,"pd3",0);
oarc_1_1("tbufs3",1,"pd3",1);
oarc_1_1("tbufs3",2,"pd3",2);
iarc_1_1("tdi3",0,"pd3",0);
iarc_1_1("tdi3",1,"pd3",1);
iarc_1_1("tdi3",2,"pd3",2);
oarc_1_1("tdi3",0,"psf3",0);
oarc_1_1("tdi3",1,"psf3",0);
oarc_1_1("tdi3",2,"psf3",0);

/* marking dependent enabling functions */

count1=0;count2=0;count3=0;

for (i=0;i<=2;i++){
enabling_1("tbufs",i,buffr_choose);
}
```

```
for (i=0;i<=2;i++){
enabling_1("tbufs2",i,buffr_choose2);
}

for (i=0;i<=2;i++){
enabling_1("tbufs3",i,buffr_choose3);
}

for (i=0;i<=2;i++){
enabling_2("tui",i,0,upstream_choose);
}

for (i=0;i<=2;i++){
enabling_2("tui2",i,0,upstream_choose2);
}

for (i=0;i<=2;i++){
enabling_2("tui3",i,0,upstream_choose3);
}

}

reward_type umc() { return(mark_1("pcomm",1));}
reward_type dmc() { return(mark_1("pcomm",2));}
reward_type pmc() { return(mark_1("psf",2));}
reward_type umc2() { return(mark_1("pcomm2",1));}
reward_type dmc2() { return(mark_1("pcomm2",2));}
reward_type umc3() { return(mark_1("pcomm3",1));}
reward_type dmc3() { return(mark_1("pcomm3",2));}

reward_type p1u1() { return(mark_1("pu",0)*5);}
reward_type p1u2() { return(mark_1("pu",1)*5);}
reward_type p1u3() { return(mark_1("pu",2)*5);}
reward_type p1d1() { return(mark_1("pd",0)*5);}
reward_type p1d2() { return(mark_1("pd",1)*5);}
reward_type p1d3() { return(mark_1("pd",2)*5);}

reward_type p2u1() { return(mark_1("pu2",0)*26);}
reward_type p2u2() { return(mark_1("pu2",1)*26);}
```

```
reward_type p2u3() { return(mark_1("pu2",2)*26);}
reward_type p2d1() { return(mark_1("pd2",0)*26);}
reward_type p2d2() { return(mark_1("pd2",1)*26);}
reward_type p2d3() { return(mark_1("pd2",2)*26);}

reward_type p3u1() { return(mark_1("pu3",0)*13);}
reward_type p3u2() { return(mark_1("pu3",1)*13);}
reward_type p3u3() { return(mark_1("pu3",2)*13);}
reward_type p3d1() { return(mark_1("pd3",0)*13);}
reward_type p3d2() { return(mark_1("pd3",1)*13);}
reward_type p3d3() { return(mark_1("pd3",2)*13);}

ac_final() {
float u0,u1,u2,d0,d1,d2;
FILE *fp1,*fp2,*fp;
int const;
double lamda2;
double time_point;

fp=fopen("data3","r");
fscanf(fp,"%d",&lamda2);

pr_mc_info();
pr_std_average();
const=10000;

/* output data to file */

fp1 =fopen ("data5.txt","a");
fp2 =fopen ("da_heu6.txt","a");

fprintf(fp1,"\n");
fprintf(fp2,"\n");

for (time_point=0.0;time_point<=1.0;time_point=time_point+0.1){
time_value(time_point);


u0=expected(p1u1);u1=expected(p1u2);u2=expected(p1u3);
```

```
d0=expected(p1d1);d1=expected(p1d2);d2=expected(p1d3);
fprintf(fp2," %f  %f  %f  %f  %f  %f ",u0,u1,u2,d0,d1,d2);

u0=expected(p2u1);u1=expected(p2u2);u2=expected(p2u3);
d0=expected(p2d1);d1=expected(p2d2);d2=expected(p2d3);
fprintf(fp2," %f  %f  %f  %f  %f  %f ",u0,u1,u2,d0,d1,d2);

u0=expected(p3u1);u1=expected(p3u2);u2=expected(p3u3);
d0=expected(p3d1);d1=expected(p3d2);d2=expected(p3d3);
fprintf(fp2," %f  %f  %f  %f  %f  %f\n",u0,u1,u2,d0,d1,d2);


}

for (time_point=1.0;time_point<=350.0;time_point=time_point+10.0){
time_value(time_point);


u0=expected(p1u1);u1=expected(p1u2);u2=expected(p1u3);
d0=expected(p1d1);d1=expected(p1d2);d2=expected(p1d3);
fprintf(fp2," %f  %f  %f  %f  %f  %f ",u0,u1,u2,d0,d1,d2);

u0=expected(p2u1);u1=expected(p2u2);u2=expected(p2u3);
d0=expected(p2d1);d1=expected(p2d2);d2=expected(p2d3);
fprintf(fp2," %f  %f  %f  %f  %f  %f ",u0,u1,u2,d0,d1,d2);

u0=expected(p3u1);u1=expected(p3u2);u2=expected(p3u3);
d0=expected(p3d1);d1=expected(p3d2);d2=expected(p3d3);
fprintf(fp2," %f  %f  %f  %f  %f  %f\n",u0,u1,u2,d0,d1,d2);

}

q=0;
pr_time_avg_expected("Jobs waiting",umc);
pr_mtta("Mean time to absorption");

fclose(fp1); fclose(fp2);

}
```

```
assert() {
return (RES_NOERR);
}

ac_init() {
fprintf(stdout,"\n Petri Nets for FMS\n\n");
pr_net_info();
}

ac_reach() {
fprintf(stdout,"\n The reachability graph has been generated \n\n");
fprintf(stdout,"\n The number of markings are  %i\n",sh);
pr_rg_info();
}
```

VITA AUCTORIS

NAME:	Shubhabrata Biswas
PLACE OF BIRTH:	Calcutta, India
DATE OF BIRTH:	May 2, 1970
EDUCATION:	Bachelor of Engineering in Mechanical Engineering, Regional Engineering College, Durgapur, India, 1988-1992.
	Master of Applied Science in Manufacturing Systems Engineering, University of Windsor, Ontario, Canada, 1993-1995.